Runtime Detection of Adversarial Attacks in AI Accelerators Using Performance Counters

Habibur Rahaman, Atri Chatterjee, and Swarup Bhunia
School of Electrical and Computer Engineering, University of Florida
{rahaman.habibur, a.chatterjee}@ufl.edu, swarup@ece.ufl.edu

Abstract—Rapid adoption of AI technologies raises several major security concerns, including the risks of adversarial perturbations, which threaten the confidentiality and integrity of AI applications. Protecting AI hardware from misuse and diverse security threats is a challenging task. To address this challenge, we propose SAMURAI, a novel framework for safeguarding against malicious usage of AI hardware and its resilience to attacks. SAMURAI introduces an AI Performance Counter (APC) for tracking dynamic behavior of an AI model coupled with an on-chip Machine Learning (ML) analysis engine, known as TANTO (*Trained Anomaly Inspection Through Trace Observation*). APC records the runtime profile of the low-level hardware events of different AI operations. Subsequently, the summary information recorded by the APC is processed by TANTO to efficiently identify potential security breaches and ensure secure, responsible use of AI. SAMURAI enables real-time detection of security threats and misuse without relying on traditional software-based solutions that require model integration.

Experimental results demonstrate that SAMURAI achieves up to 97% accuracy in detecting adversarial attacks with moderate overhead on various AI models, significantly outperforming conventional software-based approaches. It enhances security and regulatory compliance, providing a comprehensive solution for safeguarding AI against emergent threats.

Keywords—AI Accelerators, Neural Network, AI Performance Counters (APC), Adversarial Attacks, Runtime Attack Detection, Online Monitoring, AI Regulations, Malicious Usage, AI Safety, ML Engine

I. INTRODUCTION

Artificial intelligence (AI) hardware cores, known as AI accelerators, often integrated into modern System-on-Chip (SoC) architecture, provide enhanced performance for AI and machine learning tasks, and facilitate the rapid development and deployment of AI solutions. However, the widespread application of AI also presents several major attack vectors that target its inherent vulnerabilities, since AI often handles sensitive and critical information [1]–[3]. To ensure security, AI hardware manufacturers must regulate chip usage in terms of duration, operations, and user access, preventing misuse that could lead to cyberattacks, defects, or unauthorized deployment in the cloud [4]. Monitoring AI chip behavior is crucial for maintaining secure and safe operations and protecting against potential security breaches and safety hazards.

Generally, the performance and security analysis of AI chips is primarily performed at the software level, which cannot proactively prevent misuse and security threats. They cannot enforce restrictions at the hardware level, which is crucial for preventing unauthorized access/misuse and addressing the security issue of AI cores. Again, the security and usage analysis of AI chips is monitored by recording the occurrence of different metrics of the processor [5] by the hardware performance counter (HPC) [6] built in with the processor.

HPCs consisting of several numbers of counters count these events occurring within the processor and monitor general system performance metrics of threat detection in AI.

Various malicious attacks [7]–[13] on neural networks (NN) target vulnerabilities in models. Among the main attack models, the percentage occurrence of adversarial attacks is around 36%, which is the highest, causing increasing concern in the models. Although DNNs show strong performance in classification tasks, they are highly vulnerable to adversarial perturbations of the data [8]. Even small and imperceptible changes to the input data can cause the models to misclassify, exposing a critical weakness in their stability.
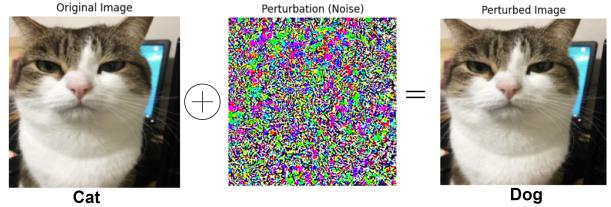

Figure 1: An example of deep-fool attack on an ML model.

An example of DeepFool adversarial perturbations is illustrated in Figure 1. The original image is classified as "Cat" and the third image incorrectly identifies "Cat" as "Dog" due to addition of small perturbation (second image) caused by the DeepFool attack, although difference between original image and adversarial image cannot be detected by human eyes.
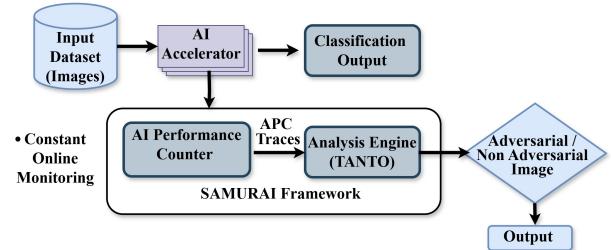

Figure 2: Block Diagram of the SAMURAI Framework illustrating the integration of the AI Performance Counter (APC) and the TANTO analysis engine, where APC captures AI hardware activity and TANTO processes the data in real time to detect adversarial attacks and unauthorized usage.

The existing techniques do not effectively monitor AI operations and detect AI-specific threats, as these do not capture AI-specific metrics like tensor operations, layer details, or specific

AI parameters. As HPCs typically monitor basic system-level metrics, they may not reflect the anomalies specific to AI operations. These methods are not effective for detecting different attacks, mainly adversarial attacks. The DeepFool attack [8], as illustrated in Figure 1, efficiently computes adversarial perturbations to fool the ML model into making incorrect predictions. This illustration strongly signifies the need for the development of an effective detection mechanism to safeguard AI models against this type of adversarial attack.

With the above motivation, for robust AI security analysis, we introduce **SAMURAI** (**S**afeguarding **A**gainst **M**alicious **U**sage and **R**esilience of **AI**), a comprehensive framework designed to protect AI accelerators against some of the most significant security threats. SAMURAI employs a multilayered low-cost security approach driven by real-time monitoring and anomaly detection techniques. The block diagram of this framework is shown in Figure 2.

This framework integrates the AI Performance Counter (APC), special purpose registers similar to the HPC in a processor, into an AI accelerator and an on-chip ML-based analyzer, known as *TANTO*. APC records the operational traces of AI, which are then processed by the *TANTO* aiming to safeguard AI against misuse, ensuring the integrity and security of AI operations. The training process is conducted directly within the ML accelerator, utilizing the APC traces. This on-device training eliminates the need to transfer APC log files (data) to a central server or cloud, thus maintaining data integrity and security. As a result, there is no requirement for online data transfer for training or decision making (on APC data). SAMURAI outperforms traditional software methods; enhances security and regulatory compliance of AI accelerators; and ensures proper use of AI in wide variety of applications, safeguarding AI hardware from unauthorized use and threats. In summary, we make the following major contributions in the proposed work.

- We identify the deficiencies of current software-based solutions for protecting AI accelerators. To address them, we introduce SAMURAI – a framework that integrates APC with an on-chip TANTO. SAMURAI leverages APCs to capture low-level hardware events and data patterns at internal nodes specific to AI operations.
- SAMURAI enhances attack resilience by incorporating TANTO (**T**rained **A**nomaly **I**nspection **T**hrough **O**bservation), a lightweight ML model, which performs real-time anomaly detection on AI operations to prevent malicious activities. This method ensures precise monitoring at minimal overhead.
- TANTO provides real-time on-device training and inferencing, eliminating the need for data transfer to a central server. This maintains data integrity and ensures efficient detection of adversarial activities in AI applications.
- Through extensive experiments on benchmark data and open-source AI models, we demonstrate that SAMURAI supports continuous and adaptive monitoring to enable robust protection against these threats without substantial performance degradation.

The remainder of the paper is structured as follows. Section II covers the preliminaries and reviews related work. Section III presents the methodology. Section IV explains the exper-imental findings and discussions. Finally, Section V presents the conclusions and future research directions.

II. BACKGROUND AND RELATED WORKS

Various attacks on neural network models can be broadly side channel attacks, fault injection attacks, trojan insertion attacks, adversarial input attacks etc, [7]–[13].

Threat Model: In this work, the performance of AI architecture is analyzed under both adversarial data generated by the adversarial attack and non-adversarial or clean data. We use an important adversarial attack known as the DeepFool attack [8] for the validation of the proposed scheme. DeepFool attack iteratively pushes the input data across the decision boundary of the classifier, making the perturbation as small as possible to remain imperceptible to humans. An adversarial perturbation is defined as the minimal perturbation p that alters the classifier's label $\hat{t}(a)$ for a given input as given below.

$$\Delta(a; \hat{t}) := \min_p ||p||_2 \quad subject \quad to \quad \hat{t}(a + p) \neq \hat{t}(a) \quad (1)$$

where a is an image and $\hat{t}(a)$ is the estimated label. We call $\Delta(a; \hat{t})$ the robustness of $\hat{t}$ at the point a. The robustness of classifier $\hat{t}$ is then defined as

$$\rho_{adv}(\hat{t}) = E_a \frac{\Delta(a; \hat{t})}{||a||_2} \quad (2)$$

where E_a is the expectation over the distribution of data.

Each type of attack requires different defense mechanisms.

AI Accelerator parameters extracted by the APC: The architectural traces of AI core are captured in real-time by APC. These values are subsequently processed by TANTO unit of SAMURAI. The AI metrics are shown in the table I.

Table I: AI accelerator metrics extracted by APC

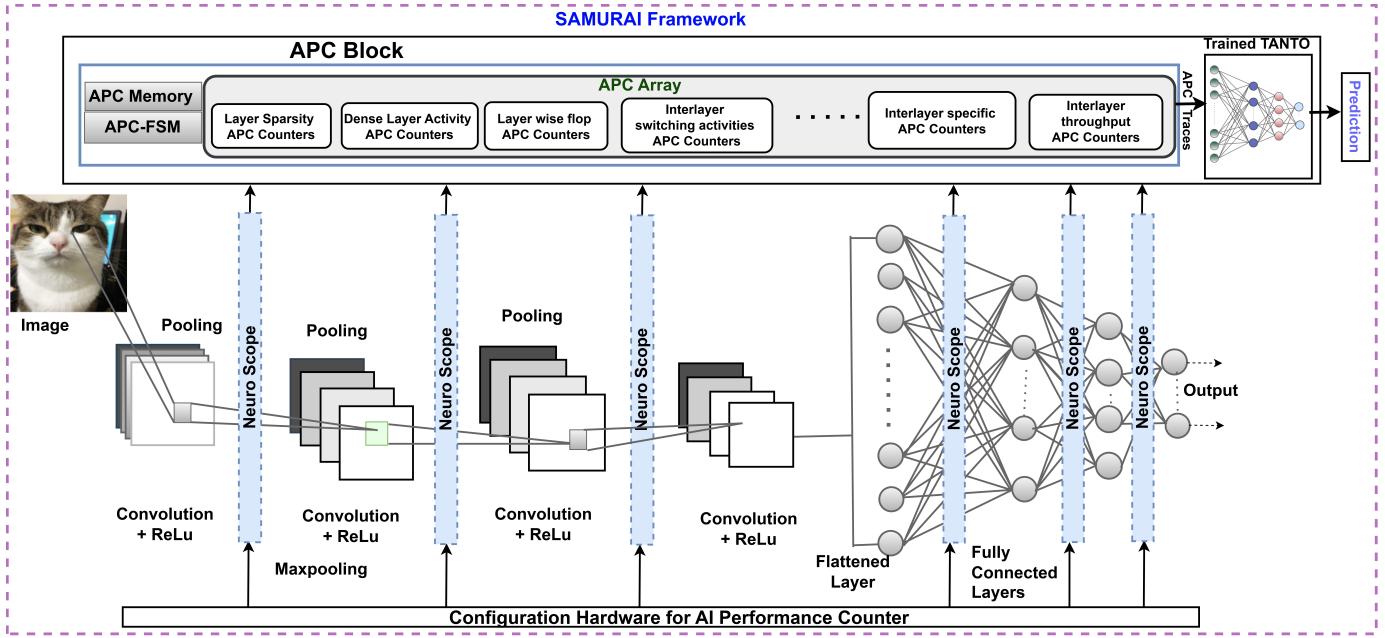| Metric	Description and Equation
Layer Sparsity	Ratio of zero-value elements in a layer's activation. $S_L = \frac{\#\{o_i \in O \mid o_i = 0\}}{\#\{o_i \in O\}}$ o_i is the i-th element in the output tensor O, and $\#\{\cdot\}$ denotes the cardinality of the set, L denotes the neural network layer with output tensor O, S_L is the layer sparsity.
Layer Zero Counts	It is the absolute count of zero- valued elements within each layer's activation. $Z_L = \sum_{i=1}^{N} \delta(o_i = 0)$ N is the total number of elements in O, o_i is the i-th element of O, and $\delta(\cdot)$ is the indicator function.
Dense Layer Activities	Includes average activation, sparsity, and extreme values. **Average Activation:** $\bar{a}_L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij}$ **Sparsity:** $S_L = \frac{\#\{a_{ij} \in A \mid a_{ij} = 0\}}{\#\{a_{ij} \in A\}}$ **Max/Min Activation:** $a_{\max} = \max(A), a_{\min} = \min(A)$ $\bar{a}_L$ is the average activation, S_L is the sparsity, $a_{\max}$ and $a_{\min}$ are the maximum and minimum activations, A is the activation matrix, N is the number of inputs, and M is the number of neurons in the dense layer.
FLOPs per Layer	Total floating-point computations per layer.
Layer-wise TOPs	Tera Operations Per Second per layer, scaled FLOPs.
Layer-wise MACs	Multiply-Accumulate operations per output element.
Layer-wise FLOPs	Individual FLOPs assessment per layer.
Layer-wise Entropies	Entropy of output distribution. $H(A) = -\sum_{i=1}^{N} P(a_i) \log P(a_i)$ $H(A)$ is the entropy of the output distribution, $P(a_i)$ is the probability of the i-th outcome, and N is the total number of classes or distinct outcomes.
Throughput	Inverse of inference time; images processed per second.

Figure 3: AI Performance Counter (APC) in an AI Accelerator illustrating the architecture of the APC module, which consists of a Finite State Machine (FSM) and a secure APC memory block that records operational metrics during AI inference and securely transmits the data to TANTO for real-time analysis.

Related Works: The work [8] presented the DeepFool algorithm, which computes minimal perturbations that mislead classifiers, allowing for a reliable robustness assessment. A side channel attack (SCA) using HPCs to infer the structure of DNNs has been reported in [5]. The work reported in [7] highlights the vulnerability of DNNs to adversarial attacks and offers an approach to the detection of attacks on networks. A detection method 'Uniguard' proposed in [14] safeguards FPGA-based AI accelerators using power side channel information generated during inference to spot anomalies. The work [9] highlights the techniques for detecting mainly SCA and Fault Injection attacks in the DNN models and AI Accelerator to enhance the security of AI accelerator. These works highlight the security analysis using software tools.

Existing software-based solutions target mainly one specific attack and require implementation into the model, making them less practical. The security analysis in AI by examining HPC traces of the processor based on AI operation faces challenges in profiling low-level hardware events of AI. The proposed framework is more integrated, straightforward, and versatile with fewer setup complexity.

III. METHODOLOGY

In the proposed security framework, the APC meticulously records low-level AI metrics during both the inference and training phases. These metrics are subsequently analyzed by an on-chip *TANTO* in real time to detect and mitigate potential security threats, such as adversarial attacks or unauthorized usage of the AI accelerator. This approach not only enhances the security of AI accelerators by providing robust defense mechanisms against malicious activities, but also ensures efficient utilization of computational resources, making it a comprehensive solution for safeguarding AI in critical appli-

cations. SAMURAI comprises an APC block and a trained ML Analysis Engine known as TANTO as shown in 3. Next, we discuss the two basic operations of SAMURAI.

A. *AI Performance Counter (APC)*

AI Performance Counter (APC) is integrated with the AI accelerator, as illustrated in Figure 3, and is designed to collect a diverse set of operational parameters. The APC block consists of an APC array, a Finite State Machine (FSM) block, and a secure APC memory block. This setup captures the key performance metrics, as detailed in Table I, during inference. To provide granular control over performance monitoring, the *configuration hardware for APC* allows dynamic selection of which counters to activate during execution. This flexibility enables users to tailor the monitoring process based on specific AI workloads, optimizing both resource utilization and security analysis. The configuration hardware facilitates the activation or deactivation of the APC counters to collect only the most relevant performance metrics, minimizing overhead while maximizing insight. Through Neuro Scope, the traces of the neural network are extracted in real time. These extracted traces are hooked into the APC block and captured into a set of specialized APCs, collectively known as the *APC Array*. The extracted data are securely stored within the *APC Memory Block*, maintaining both integrity and confidentiality. Furthermore, the APC framework incorporates a *hooking mechanism* that enables real-time interception of the execution of AI traces. This mechanism ensures that critical data points such as layer sparsity, floating-point operations (FLOPs), and memory access patterns—are captured efficiently without disrupting the normal execution flow of the AI model. Once recorded, these performance metrics are transmitted to *SAMURAI's TANTO module* via a secure communication channel, as shown in

Figure 2. TANTO processes these traces to detect potential adversarial attacks and predict security threats in real time, ensuring robust protection against AI misuse and vulnerabilities. This dynamic and configurable approach makes the AI Performance Counter a powerful tool to monitor AI accelerator behavior, detect anomalies, and protect AI hardware against adversarial threats.

B. ML Analysis Engine (TANTO)

In this framework, the data collected by the APC is transferred to the ML Engine (*TANTO*). Subsequently, the *TANTO* determines whether the operations executed by the AI accelerator exhibit unauthorized or adversarial intent. It also provides continuous feedback to the AI Accelerator both in the training and inference phase, aimed at curtailing the operations to achieve expedited convergence as well as to effectuate a reduction in power consumption. Basically, *TANTO* monitors the workloads on AI accelerators. The approach identifies and mitigates any potentially malicious activity within the workloads and alerts the user accordingly. Additionally, the AI IP owner can restrict the usability or specific inferencing capabilities of trained models.

In the next section, we discuss the operations of the proposed scheme.

C. TANTO Training Process

The TANTO training process is illustrated in Figure 4. The AI Accelerator performs inferencing on both clean and adversarial data using trained neural networks. Adversarial images are generated using the *DeepFool attack*. During this process, the AI Performance Counter (APC) captures operational traces for both normal and adversarial conditions. These traces contain key metrics such as layer sparsity and activation patterns as mentioned in table I.

To construct the training dataset, APC metrics from both clean and adversarial inputs are combined. This dataset is then used to train the machine learning-based engine, TANTO. The model training phase utilizes standard IMAGE datasets including CIFAR-10, CIFAR-100,and MNIST with widely used CNN models such as ResNet, VGG, and AlexNet. The captured features are used to train TANTO, enabling it to distinguish between original and adversarial input based on APC-derived metrics.

Once trained, TANTO is deployed in edge devices for real-time adversarial attack detection. During inference, it continuously monitors the APC traces and identifies deviations from the expected behavior. When adversarial patterns are detected, TANTO triggers security measures such as issuing user alerts or stopping compromised operations, ensuring the integrity of AI systems. The following algorithms describe the process flow step by step.

The **Algorithm 1** involves training a neural network model using a standard training dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$,where $\mathbf{x}_i$ represents the input images and y_i represents the corresponding labels. The model M is initialized with parameters θ, a loss function $\mathcal{L}$, a learning rate η, and an optimizer opt. During training, the model parameters are iteratively updated to minimize the loss function over multiple epochs E.

Once the model is trained, then **Algorithm 2** involves extracting APC metrics from the test dataset

---

Algorithm 1 Train_Model

1: **procedure** TRAIN_MODEL
2:     **Input:** Training dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
3:     **Initialize:** Neural network M with parameters θ,
       loss function $\mathcal{L}$, learning rate η, and optimizer opt
4:     **for** epoch = 1 **to** E **do**
5:         **for** each $(\mathbf{x}, y) \in \mathcal{D}_{\text{train}}$ **do**
6:             $\hat{y} = M(\mathbf{x}; \theta)$
7:             $J = \mathcal{L}(\hat{y}, y)$
8:             $\theta \leftarrow \theta - \eta \nabla_\theta J$
9:         **end for**
10:     **end for**
11:     **Output:** Trained model M_{trained}
12: **end procedure**

$\{\mathbf{APC}_{\text{original}}(\mathbf{x}_i)\}_{i=1}^{M}$. Each test image $\mathbf{x}$ is passed through the trained model M_{trained}, and internal activations (APC metrics) are captured using hook functions. These metrics provide a detailed representation of the model's internal state and are crucial for detecting adversarial perturbations. These images are marked as non-adversarial, forming the set of original APC metrics $\{\mathbf{APC}_{\text{original}}(\mathbf{x}_i)\}_{i=1}^{M}$.

Algorithm 2 Extract_APC_Metrics

1: **procedure** EXTRACT_APC_METRICS
2:     **Input:** Trained model M_{trained}, test dataset $\mathcal{D}_{\text{test}} = \{\mathbf{x}_i\}_{i=1}^{M}$
3:     **for** each $\mathbf{x} \in \mathcal{D}_{\text{test}}$ **do**
4:         $\hat{y} = M_{\text{trained}}(\mathbf{x})$
5:         Extract APC metrics: $\mathbf{APC}_{\text{original}}(\mathbf{x}) = \text{hook}(M_{\text{trained}}, \mathbf{x})$
6:         Mark image $\mathbf{x}$ as non-adversarial
7:     **end for**
8:     **Output:** APC metrics $\{\mathbf{APC}_{\text{original}}(\mathbf{x}_i)\}_{i=1}^{M}$
9: **end procedure**

Algorithm 3 DeepFool_Attack

1: **procedure** DEEPFOOL_ATTACK
2:     **Input:** Trained model M_{trained}, test dataset $\mathcal{D}_{\text{test}} = \{\mathbf{x}_i\}_{i=1}^{M}$
3:     **for** each $\mathbf{x} \in \mathcal{D}_{\text{test}}$ **do**
4:         Generate adversarial example:
            $\mathbf{x}_{\text{adv}} = \text{DeepFool}(M_{\text{trained}}, \mathbf{x})$
5:         $\hat{y}_{\text{adv}} = M_{\text{trained}}(\mathbf{x}_{\text{adv}})$
6:         Extract APC metrics:
            $\mathbf{APC}_{\text{adv}}(\mathbf{x}_{\text{adv}}) = \text{hook}(M_{\text{trained}}, \mathbf{x}_{\text{adv}})$
7:     **end for**
8:     **Output:** APC metrics $\{\mathbf{APC}_{\text{adv}}(\mathbf{x}_{\text{adv},i})\}_{i=1}^{M}$
9: **end procedure**

Next, in **Algorithm 3**, the adversarial perturbations are generated from the test data set using the DeepFool attack, which iteratively perturbs the input images $\mathbf{x}$ to create adversarial images $\mathbf{x}_{\text{adv}}$. These adversarial images are then processed through the trained model M_{trained} to extract their APC metrics. This results in a set of adversarial APC metrics $\{\mathbf{APC}_{\text{adv}}(\mathbf{x}_{\text{adv},i})\}_{i=1}^{M}$.

The **Algorithm 4** shows that the original and adversarial APC metrics are combined to build a new dataset $\mathcal{D}_{\text{APC}} = \{(\mathbf{APC}_{\text{original},i}, 0), (\mathbf{APC}_{\text{adv},i}, 1)\}_{i=1}^{M}$. This data set is used to train a machine learning-based detector model D_{ML}, which learns to distinguish between original and adversarial images based on APC metrics.
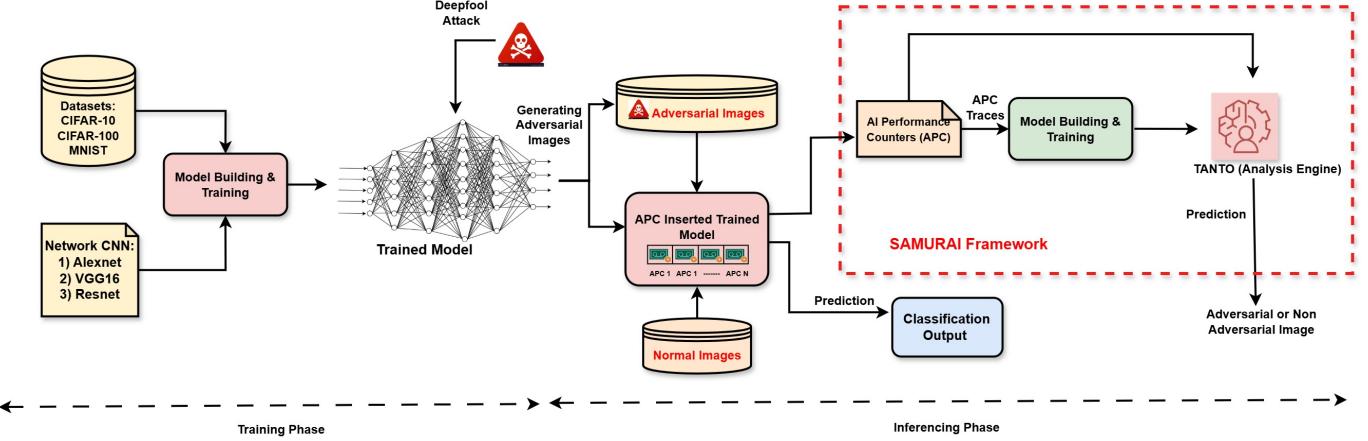
Figure 4: Adversarial Image Detection Process in SAMURAI showing how the SAMURAI framework detects adversarial images by collecting AI Performance Counter (APC) traces, analyzing them in real time, and identifying adversarial inputs during inference.

Algorithm 4 Train_Detector_Model

1: **procedure** TRAIN_DETECTOR_MODEL
2:    **Input:** APC metrics from original and adversarial images $\{\mathbf{APC}_{\text{original}}, \mathbf{APC}_{\text{adv}}\}$
3:    Construct dataset:

$$\mathcal{D}_{\text{APC}} = \{(\mathbf{APC}_{\text{original},i}, 0), (\mathbf{APC}_{\text{adv},i}, 1)\}_{i=1}^{M}$$

4:    Train machine learning model D_{ML} on $\mathcal{D}_{\text{APC}}$
5:    **Output:** Trained detector model (TANTO) $D_{\text{ML,trained}}$
6: **end procedure**

D. Monitoring and Attack Detection

Figure 4 illustrates the adversarial input detection methodology within the SAMURAI framework, describing the workflow designed to safeguard AI systems against adversarial attacks. The APCs of AI Accelerator are combined with trained TANTO, forming the SAMURAI framework which is integrated with the AI Accelerators, as shown in Figure 4. The process uses image input data sets such as CIFAR-10, CIFAR-100, MNIST and trained neural network (CNN) models such as ResNet, VGG, and AlexNet. The figure provides a step-by-step depiction of the detection process, ensuring robust security and resilience in AI-driven applications.

During inference of AI Accelerator using unlabeled adversarial or nonadversarial Image input, APCs track and record key AI operations metrics of AI operations as shown in Table I to monitor model behavior under both clean and adversarial conditions. APC generates secure APC tracers for further analysis, which TANTO examines to identify adversarial patterns by comparing operational differences between clean and adversarial data. During inference of TANTO, TANTO continuously monitors APC traces to detect deviations from expected behavior, enabling real-time adversarial detection. Upon detection, the system triggers security responses, such as alerts or stopping operations. Figure 4 illustrates this integrated approach using hardware-level monitoring and machine learning to improve AI security against adversarial threats.

By analyzing the traces during the process, TANTO can detect whether the image presented to the model is adversarial or not. The proposed framework for adversarial image detection and classification involves a multistage process aimed at enhancing the robustness of neural network models against adversarial attacks. Algorithm 5 describes the process flow for attack detection.

Algorithm 5 Infer_And_Detect

1: **procedure** INFER_AND_DETECT
2:    **Input:** Trained model M_{trained}, trained detector model $D_{\text{ML,trained}}$, workload dataset $\mathcal{D}_{\text{workload}}$
3:    **for** each $\mathbf{x}_{\text{workload}} \in \mathcal{D}_{\text{workload}}$ **do**
4:      $\hat{y}_{\text{workload}} = M_{\text{trained}}(\mathbf{x}_{\text{workload}})$
5:      Extract APC metrics:
     $\mathbf{APC}_{\text{workload}}(\mathbf{x}_{\text{workload}}) = \text{hook}(M_{\text{trained}}, \mathbf{x}_{\text{workload}})$
6:      Classify $\mathbf{x}_{\text{workload}}$ using $D_{\text{ML,trained}}$:

$$\text{is\_adv} = D_{\text{ML,trained}}(\mathbf{APC}_{\text{workload}})$$

7:    **end for**
8:    **Output:** Classification results for workload dataset
9: **end procedure**

The **Algorithm 5** shows that during the inference phase, new images of a workload dataset $\mathcal{D}_{\text{workload}}$ are processed through the trained model M_{trained} to extract APC metrics. These metrics are then classified by the trained detector model $D_{\text{ML,trained}}$ to determine if the images are adversarial. The detector model outputs a classification that indicates whether each image is adversarial or not. This comprehensive approach enhances the security and robustness of neural network models against adversarial attacks by leveraging the detailed internal activations captured in the APC metrics.

In the experiment section, the results are analyzed to verify the performance of the AI accelerator.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

We use DeepFool Attack [8] to validate the proposed technique.

Experimental Inputs:

The neural networks, datasets, and models used in the works are given below.
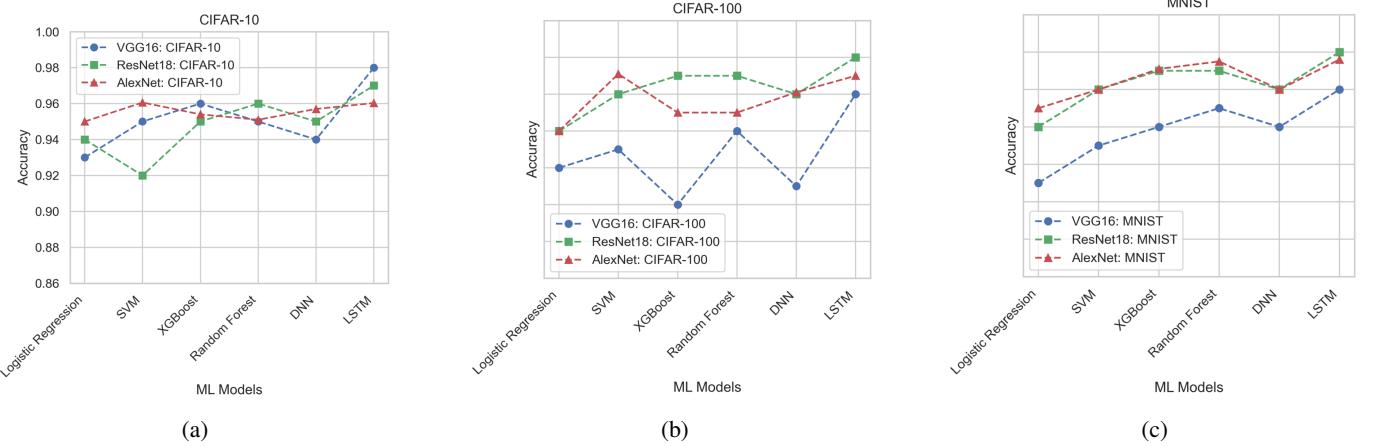
Figure 5: Comparison of Different ML Models for Adversarial Image Detection presenting the accuracy of various machine learning models, including Logistic Regression, SVM, XGBoost, Random Forest, DNN, and LSTM, in distinguishing between adversarial and non-adversarial inputs.

- **Neural Network:**
  (a) Vgg16, (b) Resnet18, (c) Alexnet
- **Dataset:**
  (a) CIFAR-10, (b) CIFAR-100, (c) MNIST
- **ML Models used:**
  Logistic Regression, SVM, XGBoost, Random Forest, DNN, and LSTM

During the inference process, the APC traces for both adversarial and non-adversarial instances are recorded and are subsequently used to train the *TANTO*. Following training, *TANTO* determines whether an image input to the AI accelerator is adversarial or not. The flow of the ML model with the APC parameter to detect anomalies in the data is shown in Figure 4.

The results of the Deepfool attack performed on different neural network architectures based on different data sets are presented in Table II. Table II shows the performance metrics of the Logistic Regression, SVM, XGBoost, Random Forest, DNN, and LSTM models in detecting adversarial perturbations, and all ML models achieve significant detection accuracy (more than 90%). Table II indicates that the LSTM model consistently achieves the highest accuracy (0.98 for CIFAR-10 and ResNet18) establishing its robustness over the different data distributions and complexities. These findings highlight that LSTM models are particularly effective in utilizing APC metrics for adversarial detection, due to their ability to capture.

Figure 5 compares the performance of various models in detecting adversarial inputs using APC metrics. The graph shows that the LSTM models consistently achieve the highest accuracy in different datasets and different neural network architectures. For example, the LSTM model achieves an accuracy of 0.98 in CIFAR-10 with ResNet18, indicating its robustness in handling diverse data distributions. The figure highlights the effectiveness of LSTM models in utilizing APC metrics for adversarial detection, underscoring their ability to capture the intricate patterns within the data that contribute to their superior performance.

Table III outlines the computational data on overhead and inference time for different models and datasets with and

Table II: **Comparison of different ML Models for Prediction of Adversarial or Non-Adversarial images**

Network	Dataset	APC Model Traces file in (MB)	Logistic Regression	SVM	XGBoost	Random Forest	DNN	LSTM
			Comparison of different ML Models for Prediction of Adversarial or Non-Adversarial					
Vgg16	CIFAR-10	280 MB	0.93	0.95	0.96	0.95	0.94	0.98
	CIFAR-100	300 MB	0.92	0.93	0.9	0.94	0.91	0.96
	MNIST	119.8MB	0.91	0.93	0.94	0.95	0.94	0.96
Resnet18	CIFAR-10	305 MB	0.94	0.92	0.95	0.96	0.95	0.97
	CIFAR-100	310MB	0.94	0.96	0.97	0.96	0.95	0.98
	MNIST	517 MB	0.94	0.96	0.97	0.97	0.96	0.98
Alexnet	CIFAR-10	106.6 MB	0.95	0.9606	0.954	0.951	0.957	0.9603
	CIFAR-100	106.6 MB	0.94	0.971	0.95	0.95	0.961	0.97
	MNIST	179 MB	0.95	0.96	0.971	0.975	0.96	0.976

Table III: Overhead for combined APC and TANTO over various models and datasets

Model (Architecture)	Dataset	Inference Time without APC(s) (Single Image) (s)	TANTO (ML Model) (S)	Inference Time for cobmined APC and TANTO Integration (s)	Overhead
ResNet-34	CIFAR-10	1.05	0.219	1.269	17.30%
ResNet-18	CIFAR-100	0.8	0.056	0.856	6.50%
ResNet-18	CIFAR-10	0.75	0.063	0.813	7.70%
AlexNet	CIFAR-100	0.62	0.069	0.689	10.00%
AlexNet	CIFAR-10	0.58	0.034	0.614	5.50%
AlexNet	MNIST	0.52	0.046	0.566	8.10%
VGG-16	CIFAR-10	1.25	0.165	1.415	11.70%
VGG-16	CIFAR-100	1.27	0.219	1.489	14.70%
VGG-16	MNIST	1.1	0.045	1.145	3.90%
ResNet-101	CIFAR-100	1.4	0.365	1.765	20.70%
ResNet-101	CIFAR-10	1.35	0.342	1.692	20.20%
ResNet-50	CIFAR-10	1.2	0.279	1.479	18.90%
ResNet-50	CIFAR-100	1.17	0.289	1.459	19.80%

without the APC integration. For example, the AlexNet on MNIST shows an 8.10% overhead, VGG-16 on CIFAR-100 experiences a 14.70% overhead, and the ResNet-34 on the CIFAR-10 dataset shows an overhead of 17.30%, with increase in inference time from 1.05 seconds to 1.269 seconds when APC is integrated. These values vary over the different models. This shows the trade-off between enhanced security and computational efficiency, demonstrating the feasibility of integrating APC and TANTO for adversarial detection without significantly compromising performance.

Figure 6 graphically represents the overhead results of the integrated APC and TANTO combination on different models and data sets. The figure 6 shows that the overhead varies
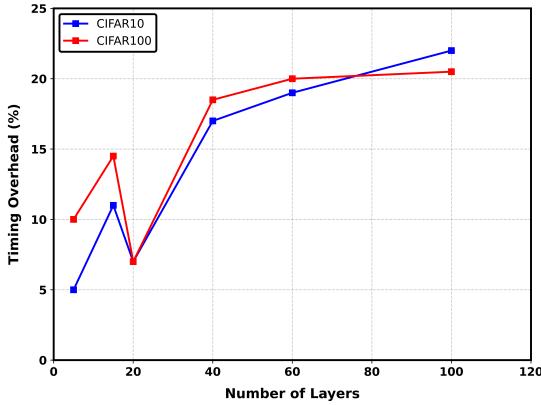
Figure 6: Timing overhead vs number of layers.

depending on the model and dataset, aligning with the data presented in Table III.

V. CONCLUSIONS

We have presented *SAMURAI* – a framework for protecting AI accelerators against major adversarial attack vectors by integrating performance counters (APC) and a runtime attack detection block called *TANTO* that acts on the recorded performance counter traces. *SAMURAI* provides enhanced security, efficiency, and integrity of AI operations at low overhead through runtime monitoring of internal events. As APC records the low-level parameter traces of AI operation, the small or insignificant perturbation, which forces the ML model to make incorrect predication is accurately detected by this technique. The on-device learning and inferencing in TANTO lead to minimal data transfer overhead, i.e., communication bandwidth while optimizing storage and processing resources, while providing real-time attack detection capability. The experimental results on various AI models using different datasets show that the proposed hardware solution has achieved approximately 97% attack detection accuracy with modest overhead. Our future work will focus on the implementation of energy and area-efficient APCs for enhanced threat detection, extension to more complex AI models, including generative AI, and scalable cloud-based security solutions.

REFERENCES

[1] Z. Kong, J. Xue, Y. Wang, L. Huang, Z. Niu, and F. Li, "A survey on adversarial attack in the age of artificial intelligence," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, p. 4907754, 2021.
[2] O. A. Osoba and W. Welser, *The risks of artificial intelligence to security and the future of work*. RAND Santa Monica, CA, 2017.
[3] T. Yigitcanlar, K. C. Desouza, L. Butler, and F. Roozkhosh, "Contributions and risks of artificial intelligence (ai) in building smarter cities: Insights from a systematic review of the literature," *Energies*, vol. 13, no. 6, p. 1473, 2020.
[4] N. Kaloudi and J. Li, "The ai-based cyber threat landscape: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–34, 2020.
[5] B. A. D. Kumar, S. C. Teja R, S. Mittal, B. Panda, and C. K. Mohan, "Inferring dnn layer-types through a hardware performance counters based side channel attack," in *Proceedings of the First International Conference on AI-ML Systems*, 2021, pp. 1–7.
[6] S. Das, J. Werner, M. Antonakakis, M. Polychronakis, and F. Monrose, "Sok: The challenges, pitfalls, and perils of using hardware performance counters for security," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 20–38.
[7] A. Madry, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
[8] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.
[9] S. Mittal, H. Gupta, and S. Srivastava, "A survey on hardware security of dnn models and accelerators," *Journal of Systems Architecture*, vol. 117, p. 102163, 2021.
[10] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitraş, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *arXiv preprint arXiv:1810.03487*, 2018.
[11] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1211–1220.
[12] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.
[13] H. Liang, E. He, Y. Zhao, Z. Jia, and H. Li, "Adversarial attack and defense: A survey," *Electronics*, vol. 11, no. 8, p. 1283, 2022.
[14] X. Yan, H. Qiu, and T. Zhang, "A unified hardware-based threat detector for ai accelerators," *arXiv preprint arXiv:2311.16684*, 2023.