EFFICIENT LEARNING FOR HARDWARE SECURITY VALIDATION USING ELECTROMAGNETIC SIDE CHANNELS

A Dissertation
Presented to
The Academic Faculty

By

Erik J. Jorgensen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Machine Learning in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2022

EFFICIENT LEARNING FOR HARDWARE SECURITY VALIDATION USING ELECTROMAGNETIC SIDE CHANNELS

Thesis committee:

Dr. Alenka Zajić, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Matthieu Bloch, Co-Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Milos Prvulovic
School of Computer Science
Georgia Institute of Technology

Dr. Mikko Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin - Madison

Date approved: August 10, 2022

For Maggy!

This thesis would not exist without the guidance and support of many people. Prof. Alenka Zajić and Prof. Milos Prvulovic took a chance bringing me into the lab and seeing my path toward a doctorate when I was still seeing only one day ahead. I am wholly appreciative for them having taken that chance and further building me into an independent researcher. Any forethought or organization in my research and writing is due largely to their influence.

I also thank my committee members who each have had great impact on my experience and direction throughout my ongoing training in research. I thank Prof. Mikko Lipasti for patiently molding my skills in research and encouraging me to aim higher than I thought made sense. I thank Prof. Mark Davenport for generating enough excitement about signal processing at Georgia Tech to make me want to move out of my comfort zone in the Midwest; and continue when it wasn't always easy. I also want to thank Prof. Matthieu Bloch and Prof. David Anderson for evolving my interests in machine learning through their teaching, mentorship, and kindness toward students.

I need to thank my labmates and friends, both current and former, for helping me stay motivated, keeping things light, and helping me work through countless silly mistakes. I never pictured myself moving to the Southeast, but thank you all for making this whole thing fun. And to Jihui for all the fun we had grading or keeping me afloat in coursework; while constantly being a sounding board for all things research, jobs, and life.

I want to thank my family for all their support and love as I took my time finding out where I'm headed. I thank my brother and sister for clearing the brush so I could find my own trails in life. I thank my parents for their support in every way imaginable, even when I decided to move half the country away. Finally, I thank Nicole for making the choice to follow me when I didn't know where we were going, under circumstances that would have turned away any person less selfless, to learn and grow together far from home.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this thesis is to combine the non-destructive monitoring advantages of standard and backscattering electromagnetic side channels with modern machine learning techniques to efficiently validate the authenticity of individual integrated circuits installed on a motherboard.

The authenticity of integrated circuits is of increasing concern as more steps in the device manufacturing supply chain are outsourced, especially in light of severe global semiconductor shortages. Common methods for integrated circuit validation rely on either destructive techniques before high resolution imaging of the circuit interconnects or functional testing of a variety of test inputs with automated test equipment. These methods are time-consuming or even intractable to detect counterfeit components or stealthy modifications of their underlying circuitry.

Side channels are any means of remotely leaking information related to a circuit's activity or architecture. Our work takes advantage of the electromagnetic (EM) side channel to remotely capture identifying information emitted from or backscattered off integrated circuits in the form of EM signals that can be used to validate their authenticity.

This research attempts to alleviate the need for time-consuming and expensive destructive validation methods for hardware security by robustly detecting inauthentic or modified integrated circuits with remote EM side-channel measurements. The first aim of this research is to apply deep learning methods to classify and detect counterfeits of major ICs on a variety of motherboards. The second aim is to leverage hyperspectral scanning with the backscattered EM side-channel and a novel active learning method to detect dormant hardware trojans several times smaller than before. The last aim is to develop a compressed sensing approach to heavily reduce sampling for hardware trojan detection as well as to develop a hyperspectral characterization of expected and anomalous circuits.

CHAPTER 1
INTRODUCTION

1.1 Motivation

As companies choose to outsource more steps in the electronic device manufacturing pipeline, the authenticity of the integrated circuits contained on those devices becomes of increasing concern. A device designer cannot trust the origins of its devices blindly due to the ever-increasing number of entities involved in their development and manufacture. A fabricated IC is vulnerable to potentially malicious actors across the design, fabrication, testing, packaging, and other stages. It is estimated that counterfeit ICs represent about 1% of semiconductor sales [1], resulting in losses in excess of $100 billion annually [2]. Recent semiconductor shortages worldwide are expected to exacerbate the counterfeiting problem. The integrated circuit (IC) supply chain involves several steps with components passing through the facilities of several companies, often in different countries, on the route from a silicon foundry to the electronic device delivered to a customer. Anywhere along that supply chain, malicious actors could introduce counterfeit or otherwise modified ICs in place of the components intended to be placed. Even more, firmware or bitstream modifications could have a similar effect even if the IC is the exact hardware expected. These suspicious IC intrusions could be as simple as the use of a cheap alternative IC posing as the real thing to cut costs, or as severe as the inclusion of hardware Trojans (Trojans or HTs) introduced at the silicon foundry covertly change the circuit's expected behavior. In applications of critical importance like for devices used in infrastructure, military, or healthcare, these inauthentic or modified ICs present a severe risk to both safety and security.

To detect these suspicious ICs, it is common to analyze components through physical or electrical inspection techniques. Thorough physical inspection typically requires destructive defacing of the chip, layer-by-layer, to capture high resolution images of internal circuitry for validation of the layout [3]. Electrical inspection techniques can involve lengthy automated tests that test a set assortment of cases that may not uncover counterfeit ICs or malicious activity on the IC [4]. Side-channel methods have received significant attention for their ability to remotely capture information leaking from devices that can be used to fingerprint them. While certain side channels have been demonstrated to fingerprint entire devices or even some modifications to the ICs on those devices, non-destructive methods are still needed to detect individual inauthentic components or covert modifications to the components.

This thesis demonstrates learning strategies to non-destructively detect counterfeit or inauthentic components at a significantly smaller scale than has been achieved in literature thus far, to the best of our knowledge. Paired with high bandwidth side-channel measurements, this research develops novel methods to detect counterfeit integrated circuits and efficiently uncover hardware Trojans at a scale not previously possible.

1.2 Deep Learning Classification of Motherboard Components by Leveraging EM Side-Channel Signals

It is well known that electronic components leak information through several different "side channels." When a digital device is powered on, this information leakage could arise as variations in acoustic noise [5], temperature [6], power consumption [7], electromagnetic (EM) emanation [8], or any other information leakage through empty space [9]. The conductive traces on a device and inside integrated circuits act as antennas when the electrical current running through them is changing. It has been shown that these changes in current can emanate significantly more information about device activity than the other side channels listed previously [10], [11]. Known as the EM side channel, this information leakage through electromagnetic waves caused by circuit activity has traditionally been cause for security concern by being exploited to steal confidential information [8], [10], [12], [13].

At the same time, these EM side-channel emissions have also been used for identifying devices from the scale of vehicles [14], [15] down to cellphones or microcontrollers [16], [17]. While these EM side-channel techniques have shown success for classifying entire devices, they have not been shown to identify individual ICs or components embedded on those devices.

There is no established state-of-the-art solution to the problem of non-destructively identifying individual components on an *assembled* motherboard from EM side-channel signals. While there has been work to classify single components or entire motherboards based on their EM signatures [18], [16], to our knowledge, there has been no work on classifying components already integrated onto a motherboard. This testing scenario is crucial for device designers, who need to validate that the components on their devices have not been replaced with counterfeits somewhere in the device supply chain before delivering their product to customers.

We present a deep learning strategy to classify components on already-assembled motherboards by leveraging their EM side-channel emanations. With this method, a variety of integrated circuit components are classified from their EM emanations using a lightweight convolutional neural network (CNN) architecture that we specifically design for this data-scarce scenario with high-dimensional measurements. Analyzing input signals as they pass through our model allows us to generate insights about the most important discriminative features for classifying each component type and model. Our results demonstrate that we can classify a diverse set of ICs accurately in a practical scenario.

1.3 Feature Selection for Non-Destructive Detection of Hardware Trojans using Hyperspectral Scanning

Hardware Trojans are malicious and unauthorized circuitry modifications made to integrated circuits. Like counterfeit components, HTs represent a great security threat that can be introduced at several stages of the device supply chain. These Trojans can be responsible for locking devices, leaking secure information, draining battery, or a host of other changes that can have devastating consequences [19], [20], [21], [22]. When embedded into circuits at the silicon foundry or packaging stages of the device supply chain, these hardware modifications can supersede secure software that relies on a trusted hardware platform. Alternatively, these Trojans may have similar effects by being loaded onto a field programmable gate array (FPGA) through an inconspicuously corrupted bitstream [23], [24]. Thus, detecting ICs whose security has been compromised by a Trojan is of critical importance before those devices have been deployed or the Trojan activated.

No matter their origin, HTs are designed to be stealthy and difficult to detect by routine automated functional testing by activating under rare circumstances [19]. A typical HT is designed with a *trigger* that monitors circuit behavior or inputs before activating a *payload* that executes whatever malicious activity the Trojan is designed to do. A dormant Trojan which has not yet been activated does not modify normal circuit behavior or draw significant power. This makes it particularly hard to detect without destructive testing methods that look at circuit trace architectures with high resolution. Non-destructive detection methods such as side-channel analysis often rely on monitoring for significant changes in current or power consumption to detect when a Trojan is activated. However, detecting a Trojan before its activation is much more valuable due to the grave concerns about the damage that they can inflict when activated.

One method, using the backscattering EM side channel, has shown some success detecting dormant HTs before their activation [25]. This backscattering HT detection method draws inspiration from radio frequency identification (RFID) systems which modulate an incoming signal to transmit information through the signal that reflects back to the transceiver [26], [27]. Similarly, this HT detection method transmits a signal at an IC and captures the reflected (backscattered) waves that will by modulated by a combination of circuit trace architecture and program activity. With large enough Trojans, the difference in transistor architecture between an uninfected and Trojan-infected circuit can be distinguished by the structure of those backscattered signals. Unfortunately, that method was only able to detect large dormant trojans and required tedious manual probing to uncover each type of Trojan's presence. Without *a priori* knowledge of the size of the Trojan or locations on an IC at which it would most reveal itself, that backscattering method may be difficult to implement in the real world.

To overcome the shortcomings of those existing non-destructive HT detection methods, we develop a novel approach to detect dormant HTs through intelligent sampling of hyperspectral backscattering EM side-channel measurements. We spatially scan the IC across space and frequency with a high-resolution probe to generate point-scanned hyperspectral images of the backscattered EM signal. We develop a novel filtering and active learning strategy to greatly reduce the large feature space and selectively capture scans of features based on the confidence of their measurements. This method improves our ability to detect dormant hardware Trojans of much smaller sizes than in prior work.

1.4 Hyperspectral Image Recovery via Reliability-Weighted Compressed Sensing for Hardware Trojan Detection

While detecting hardware Trojans using hyperspectral measurements of the backscattering EM side-channel shows tremendous improvements in being able to detect significantly smaller dormant Trojans, it also requires the trade-off of requiring additional sampling time versus methods before it. One of the main benefits of non-destructive HT detection methods is generally their higher throughput compared to destructive methods. A destructive test can involve decapsulating an IC, capturing high resolution images of the surface, removing another layer of material, and repeating the process until the entire trace architecture is imaged [3], [28]. Then, those layered images are painstakingly compared to the IC's netlist-level design by skilled scientists; an entire process which is time-consuming and completely destroys the tested IC. A non-destructive test then has the obvious advantage of not needing to destroy the device under test (DUT), but also being able to make a classification of the IC within minutes or seconds based on its response to automated inputs or its side-side channel emanations. Point-scanning hyperspectral images, while non-destructive and highly effective at detecting small dormant Trojans, can be heavily time-consuming. As we demonstrate in [29], these measurements can potentially requiring an hour or more to scan a single IC. To maintain the benefits of hyperspectral scanning but reduce measurement time, we turn to compressed sensing.

Compressed sensing (CS) is a popular framework to recover high-dimensional images from relatively few samples in a variety of problem domains. CS enables the accurate recovery of signals even when sampling far less than the Shannon-Nyquist rate, assuming the data is sparsely represented in some basis [30],[31]. In the hyperspectral domain, CS is typically used to recover two-dimensional images at visible or near-visible light frequency at tens of frequencies [32], [33]. Due to measurement hardware constraints, capturing high resolution images of the backscattering EM side channel can only be done by point-scanning; one pixel at a time. Thus, we focus on a less-common version of CS which captures individual pixels of an image, uniformly at random, and reconstructs them into full images. However, the differences in hyperspectral measurements caused by small, dormant Trojans are very sparse and not necessarily structured spatially [29]. Sparse uniform random sampling for image recovery with compressed sensing is likely to miss these defects.

To increase the likelihood of uncovering the EM disturbances caused by a dormant hardware Trojan while also maintaining the measurement reduction benefits of a compressed sensing scheme, we develop a non-uniform sampling method that weights sampling toward unreliable measurement regions. This sampling strategy maintains sampling randomness to follow the compressed sensing framework and allows the strategy to recover the backscattered EM side-channel signals more quickly and more accurately. Doing so heavily reduces measurement requirements while maintaining or improving on HT detection performance.

1.5 Research Contributions

This section outlines a summary of the main contributions of this thesis.

- A method for device designers to non-destructively validate the identity of components and detect unseen components on already-assembled devices [34].
- A novel deep learning architecture, pre-processing, and validation strategy to ensure classification robustness in high-dimensional, data-scarce scenarios using electromagnetic side-channel signals [34].
- Feature activation map modeling to compare discriminative features learned by the deep network with conventional hand-crafted spectral features [34].
- A hyperspectral point-scanning methodology for measuring IC fingerprints with the backscattering EM side channel [29].
- State-of-the-art non-destructive detection of hardware Trojans as small as 0.03% of the circuit size [29], [35].
- An active learning and feature selection approach to for capturing and comparing hyperspectral images of the backscattering EM side channel [29],
- A reliability-weighted compressed sensing technique for recovering point-scanned hyperspectral images with heavily reduced measurement requirements using up to ten times fewer measurements than previous efforts [35].

1.6 Thesis Outline

The remainder of this thesis is organized as follows. chapter 2 discusses the background of counterfeit ICs, hardware Trojans, EM side channels, hyperspectral imaging, and compressed sensing. chapter 3 discusses efforts to develop robust integrated circuit classification and counterfeit detection on assembled devices. chapter 4 details the first efforts at

Trojan detection with hyperspectral, backscattering EM side-channel signals. Chapter 5 presents a compressed sensing approach to hyperspectral imaging that attains state-of-the-art non-destructive Trojan detection performance. Finally, Chapter 6 draws conclusion to this thesis by summarizing our research contributions and discussing future areas of research.

CHAPTER 2
BACKGROUND

2.1 Counterfeit Components

A counterfeit integrated circuit is any semiconductor component that is misrepresented as an authentic version of another IC. These could be copies produced by unauthorized manufacturers, discontinued older models, recycled versions represented as new, or many other misrepresentations [36], [37]. These counterfeits may be placed in designs at many stages of the IC supply chain for intentionally malicious reasons, to lower costs, or simply due to negligence or ignorance. In fact, each third-party supplier or assembler in the device manufacturing pipeline introduces a potential vulnerability to counterfeit components being swapped in for their authentic counterparts. No matter their origin, these components can be unreliable, lead to inter-operability issues between software and hardware, and pose a safety and security risk. Components must be tested after the device's complete assembly to be sure that no counterfeits were introduced anywhere in the manufacturing pipeline.

Here we make a few assumptions about the potential threats that a counterfeit component poses. First, we assume that counterfeit components in our intended scenario have different underlying architecture. Counterfeit components with the same transistor architecture pose less of a threat since they should operate in the same way as the intended component, barring any manufacturing variabilities introduced by the counterfeiting supplier. Given a counterfeit with different underlying circuit trace or transistor architecture, the differences will result in variations of the EM side-channel emanations compared to the authentic component since the EM side channel is dependent on the antenna-like characteristics of traces. Even visually identical components with the same footprint and pinout like those shown in Figure Figure 2.1 may have transistor-level architecture differences that

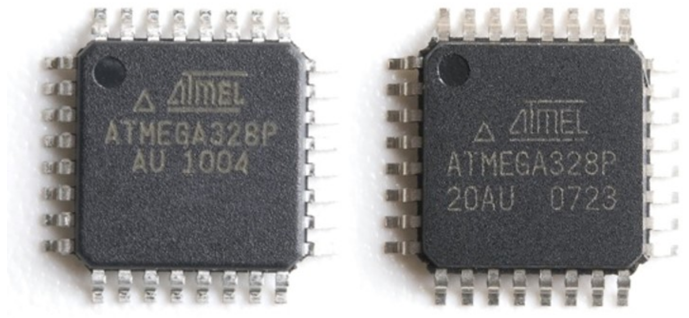


Figure 2.1: Integrated circuits with identical footprint and pinout, but a potential warning sign for counterfeiting given their distinct marking styles [38]

could result in security vulnerabilities or affect performance characteristics.

Second, we assume the device's program activity is in the same state as the devices on which the model is trained. Our method validates the authenticity of hardware through side-channel emanations that change depending on software activity. Component validation is most simple and reliable for the device designer to perform when the device is powered on in an idle state without loaded program activity. While it may be a stronger assumption to make for devices that are already deployed and in use, it is a standard assumption that the devices be tested after assembly by the device designer or another trusted entity. To ensure our method still can be useful after a device is deployed, we also test our method's performance when the device is running looping program activity to compare with prior work [39]. This looping program activity can be a good proxy for testing edge or internet of things devices that are designed to repeat the same activity monitoring or other repetitive task at all times. However, the validation that we present in this work is not necessarily possible when devices are running unknown programs because these programs can have significant effects on the EM side-channel emanations.

Finally, we assume that the device designer has possession of at least one component that could be a potential counterfeit swapped in for the original, authentic component. IC fabricators regularly iterate on their designs, so it is reasonable to assume knowledge of older IC models that pose a counterfeit risk due to their lower cost but similar design. The neural network model we present here extracts discriminating features from the EM side-channel emanations of a known set of components and devices. The feature embeddings learned by the presented CNN are used to differentiate between known component models that could be swapped in as counterfeits. Deviation from the CNN's activations for known component models is used as an indicator to detect never-before-seen counterfeits through anomaly detection. While this method could work using measurements of multiple unrelated components, training with very similar components allows the network to learn sensitive yet robust discriminative features to detect those small differences.

2.2 Hardware Trojans

At a smaller scale, components can also be compromised with hidden hardware trojans. Hardware Trojans present on a device represent a significantly more serious risk than simple design errors or natural runtime errors. While random errors may be accounted for with error correction code, HTs act covertly and adversarially which can have unforeseen effects on otherwise secure software. There has been much work to understand the threat that HTs pose [40], [21], [41] while also further understanding and characterizing their implementation [42], [43], [44], [45]. A device designer cannot trust the origins of its devices due to the number of entities involved in their development and manufacture. Much like a circuit counterfeit being replaced for the authentic chip, a fabricated IC is vulnerable to potentially malicious actors across the design, fabrication, testing, packaging, and other stages. Since all of these steps are typically completed by separate specialized entities, ICs fabricated in the most robust supply chains may be vulnerable to HT injection by bad individual, corporate, or political actors. A typical HT threat might modify the design of an application-specific integrated circuit (ASIC), digital signal processor, microprocessor, or other hardware at an untrusted foundry [20]. Even when a trusted designer securely develops circuit modules and generate their layouts using trusted design tools, sending the layouts to an untrusted foundry for manufacture leaves a window for potential modification of the circuit layouts [46], [20]. Thus, it is necessary to perform final design verification of the functional IC to detect the presence of HTs that may be added in this pipeline.

A growing body of research focuses on analyzing the threat and implementation of HTs [40], [21], [41], [42], [44], [45]. Though traditionally these Trojans are thought of as trace- or transistor-level modifications to an integrated circuit, the recent prevalence of field programmable gate arrays (FPGAs) for prototyping or specialized critical applications also presents a risk for malicious Trojans inserted as firmware modifications through the FPGA's bitstream [23], [24]. Even if an FPGA's configurable logic blocks and programmable interconnects are not modified at the hardware level, the addition of covert Trojans into the bitstream can implement malicious logic independent of the hardware platform [24]. Though the performance gap between FPGAs and ASICs is reducing, FPGAs do not necessarily have the same circuit properties as ICs [23] and further investigation of HTs inserted into ASICs is necessary to validate detection techniques. However, the path length and resulting impedance changes caused by an inserted Trojan that affect EM side-channel emanations or reflections could still be present whether in an FPGA or ASIC [23], [41]. The Trojans analyzed in this work are implemented on FPGA platforms versus custom ASICs for reasons of practicality.

These malicious modifications are designed to be triggered by rare inputs or circuit activity so that the HT can avoid being detected in routine functional testing [21]. Their activation mechanism is known as a trigger and is paired with a payload that modifies the IC's original behavior, as shown for an example circuit in Figure Figure 2.2. As further detailed in [19] and [20], we analyze these rarely-activated Trojans that use what is called a *internally-activated, condition-based* triggering mechanism. As opposed to *externally-activated* triggers that require outside interaction or *always on* triggers that are inserted on rarely-used circuitry, these triggers use small additional circuitry on the IC to monitor a program state, sensor level, input pattern, or other combinational or sequential logic condition to activate the payload. Before activation by the trigger, the dormant payload does not modify normal circuit behaviour, making it difficult to detect without thorough destructive testing. This trigger and payload design of HTs allows the larger payload circuitry to remain dormant while only a small bit of trigger circuitry is active. Small trigger circuits draw little enough current when dormant compared to the IC's normal circuitry to remain undetectable with power analysis methods. However, once triggered, that HT's payload can leak secure information, drain battery, or many other nefarious activities. The HT's malicious purpose may be functional or non-functional. A functional HT might change the value of the main circuit's outputs or communicate secure information externally while a non-functional HT may cause rapid battery drain or leak sensitive information through a side channel, among many other possibilities. Since the effects of an activated Trojan can be so devastating, it is critical to detect their presence before activation or, ideally, before the device is even deployed. Detecting a dormant Trojan has proven difficult without destructive testing methods that can uncover tiny circuit modifications from high resolution imaging. However, non-destructive methods like side-channel analysis have received more attention to develop strategies that can detect smaller and smaller Trojans.

2.3 Electromagnetic Side Channel

EM side-channel signals are emissions due to changes in current in the ICs and conductive traces on electronic devices [10], [47]. When a component on a digital device is supplied power, the current-based EM emanations from the component follow consistent patterns based on a host of factors. Among any number of noisy variations in the signal, emanations change as a result of the component's signal activation or physical properties. With repeated excitation, such as a device clock signal or repeating code pattern, we expect to see emanations that are especially noticeable in the frequency domain. These emissions can show up as sharp spikes with harmonics in the frequency domain due to consistent transistor switching from a clock signal or looping program activity [9]. While the EM side channel has proven useful for fingerprinting devices based on their emanations due

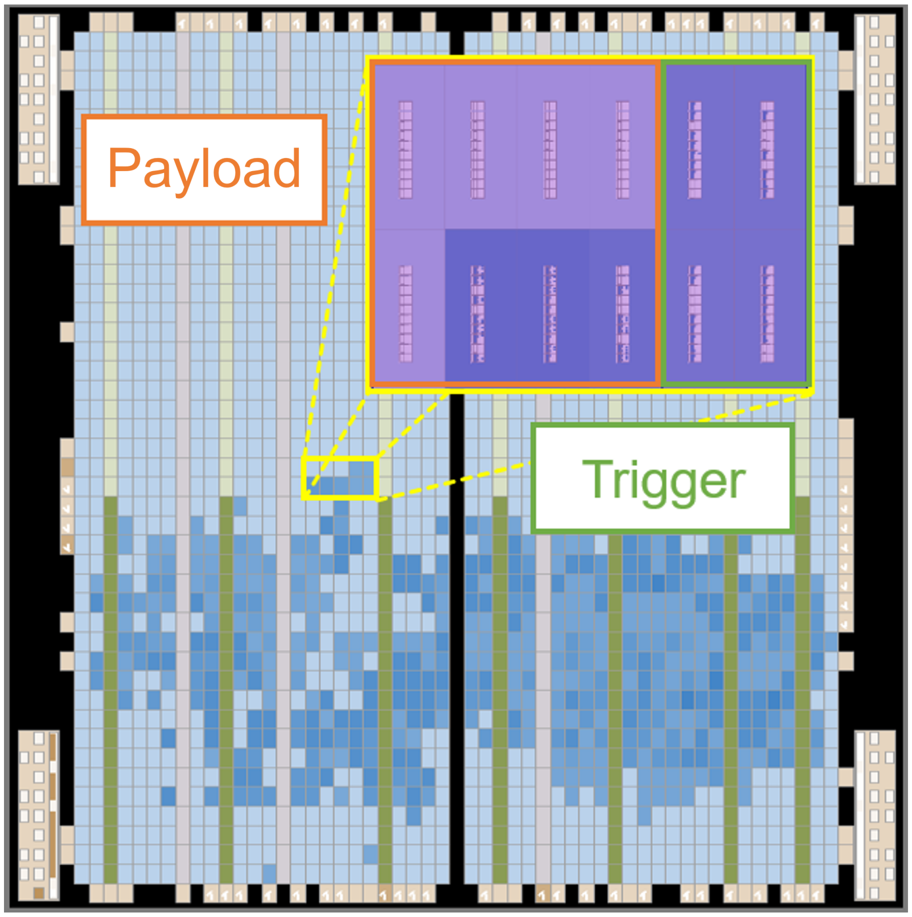


Figure 2.2: Hardware Trojan payload and trigger layout hidden in an encryption circuit. Darker shaded blue blocks represent areas using greater logic and storage resources for the circuit when loaded onto an FPGA. This circuitry takes up approximately 0.7% and 0.6% of the area of the normal circuit resources for the Trojan's trigger and payload, respectively.

to changing current passing through their components, the static or inactive properties of devices will not naturally produce emanations at a similar power scale. IC resources that lack current flow will not leak significant emissions and are essentially hidden from the perspective of the traditional EM side channel or other current-based side channels [25]. More recently, impedance-based side-channel analysis has shown the ability to fingerprint devices based on their geometric and material properties as a whole.

At a large scale, impedance-based side-channel analysis has been used to fingerprint large metal parts by their response to piezoelectric stimuli to assess their manufacturing quality [48]. Until recently, similar methods had not been developed to probe devices or components at the scale of integrated circuits. The authors of [25] developed what is known as the backscattering EM side channel to reduce issues with noise and interference with the original EM side channel. That work draws inspiration from the way that RFID tags can transmit bits by switching between two antennas with different impedance [49]. The RFID reader's transmitted signal at frequency f_t acts as a carrier for the impedance of the tag that alternates at f_c, resulting in an amplitude-modulated signal returning bits of information back to the reader. Similarly, the backscattering EM side channel is captured by transmitting a signal which is modulated depending on the state of an IC. The backscattered signal reflects back differently depending on the transistors' states, which have different impedance when connected to a voltage source versus being grounded [50]. The reflected signal is captured with an EM probe and holds information about the transistor architecture of the circuitry, even without current passing through it. The majority of transistor switching occurs at the IC's clock frequency f_c; picked up by a probe as modulated spikes in the frequency domain at $f_t \pm f_c, \pm 2f_c, \ldots$ as shown in Figure Figure 2.3. Variations of these spikes occur depending on the specific circuit architecture and activity in the region of the probe. These variations may flag the presence of a Trojan in the circuit when compared against a reference signal. Circuits infected with dormant condition-based HTs can be detected through the backscattering EM side channel because their distinct transistor ar-

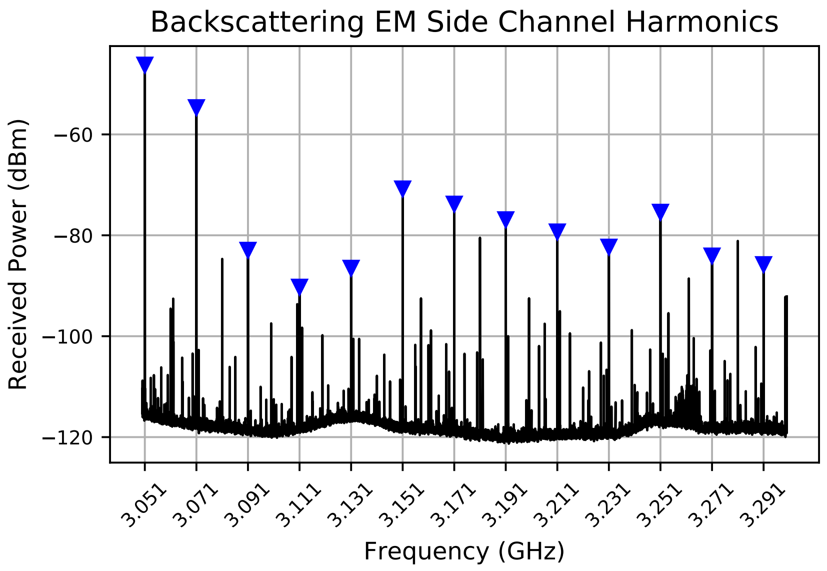


Figure 2.3: Received power of backscattered electromagnetic signal with harmonics above the transmitted frequency $f_t = 3.031$ GHz, separated by the device's clock frequency $f_c = 20$ MHz.

chitecture reflects EM signals differently than their uninfected circuit counterparts. While we do not test other types of internally-activated HTs in the following works, the addition of any Trojan that modifies the IC layout could potentially affect the backscattered signal enough to be detectable. The backscattering EM side channel has the main advantages over the original EM side channel of being impedance-based and being adjustable to avoid frequency bands with significant interference.

2.4 Hyperspectral Imaging

Hyperspectral imaging (HSI) typically refers to the collection of many two-dimensional images at different frequency bands in the visible or near-visible light spectrum . While normal color photos are represented in 3-dimensional space with two spatial dimensions and one frequency dimension for the red, green, and blue channels, HSI captures tens or hundreds of frequencies in a 3-dimensional hypercube of data per image. While visible light HSI is typically used in fields such as geoscience [51] and biomedicine [52], [53], some HSI techniques have been used to analyze the quality and composition of printed circuit boards [54], [55] for electronics recycling. HSI is typically performed in one of three ways: point-scanning, spectral scanning, and non-scanning imaging [56]. Point-scanning captures individual measurements across frequency, one at a time, and raster scanning in space to capture the whole data hypercube. Spectral scanning involves capturing an entire 2D spatial scene at a single frequency before physically switching or tuning the sensors to the next frequency and repeating. Finally, non-scanning or snapshot imaging captures the entire hypercube at once through projected acquisition and subsequent reconstruction back to the correct perspective [57]. Unfortunately, these well-researched HSI techniques with visible or near-visible light do not penetrate an IC's packaging to fingerprint an integrated circuit.

With existing high resolution probes, the backscattered EM side-channel has been shown to pass through an IC's packaging and reflect back differently based on the underlying trace architecture [58]. Given those constraints, point-scanning must be used to create non-traditional hyperspectral images of the backscattered EM side-channel signals needed to fingerprint ICs at a small scale. Unfortunately, the main drawback of point-scanning is the large increase in scanning time to capture entire hyperspectral images since the time physically move a probe is significant for high-dimensional images. On the other hand, point-scanning allows for flexibility in scans. Scans do not necessarily need to be made sequentially; left-to-right and top-to-bottom. Additionally, point-scanning allows for an arbitrary number of measurements at a fixed point in the spatio-spectral space. With this flexibility, we can investigate unique forms of measurement frameworks like compressed sensing.

2.5 Compressed Sensing

A popular way of reducing measurement costs is to reconstruct images from very few measurements in a compressed sensing framework. Unlike iteratively selecting individual measurements as in an active sampling or reinforcement learning regime, CS takes advantage of an assumed underlying structure in the data's domain to allow sparse random sampling for reconstruction [31]. Many works used basis functions like the Fourier transform or discrete cosine transform (DCT) to approximate the smoothness and continuity of the data's domain [59]. For better reconstruction of natural images, other bases were developed that incorporated piece-wise smooth properties like various wavelet functions [60]. Further work showed that an overcomplete dictionary of bases learned from images similar to those being reconstructed often outperformed any of the former bases for which closed form analytical expressions exist [61], [62].

The CS framework can be represented with the standard linear model $\mathbf{y} \approx A\mathbf{x}$ where $\mathbf{y}$ is a vector of measurements, $\mathbf{A}$ is a measurement sampling matrix and $\mathbf{x}$ is a set of learned coefficients that represent the reconstructed image. Assuming $\mathbf{x}$ can be sparsely represented by some set basis vectors $\mathbf{\Psi}$, then this problem can be rewritten as $\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{\Psi}\mathbf{s}$ where s is the sparse coefficient vector reconstructing $\mathbf{x}$ in that basis, meaning $\mathbf{x} = \mathbf{\Psi}\mathbf{s}$ as seen for an example image reconstruction problem in Figure 2.4. To ensure that the solution can be stably recovered, $\mathbf{A}$ and $\mathbf{\Psi}$ should be incoherent, meaning they cannot sparsely represent the other. In practice, $\mathbf{\Psi}$ could be the Fourier, Wavelet, learned dictionary, or any other bases that are sparsifying for the signals to be reconstructed. $\mathbf{A}$ is then incoherent with $\mathbf{\Psi}$ if it is structured for uniformly-random sampling. This binary matrix $\mathbf{A}$ could be thought of as measuring a sparse random set of pixels which are then used to reconstruct the entire 2D image. In the heavily underdetermined setting where the number of samples is far fewer than the dimensionality of the data, the sparse recovery problem shown in (Equation 2.1) has a unique solution and can be solved with a standard LASSO solver [63], [64].

$$\min_{\mathbf{s}} ||\mathbf{s}||_1 \quad \text{subject to} \quad ||A\Psi\mathbf{s} - \mathbf{y}||_2 \leq \epsilon \tag{2.1}$$

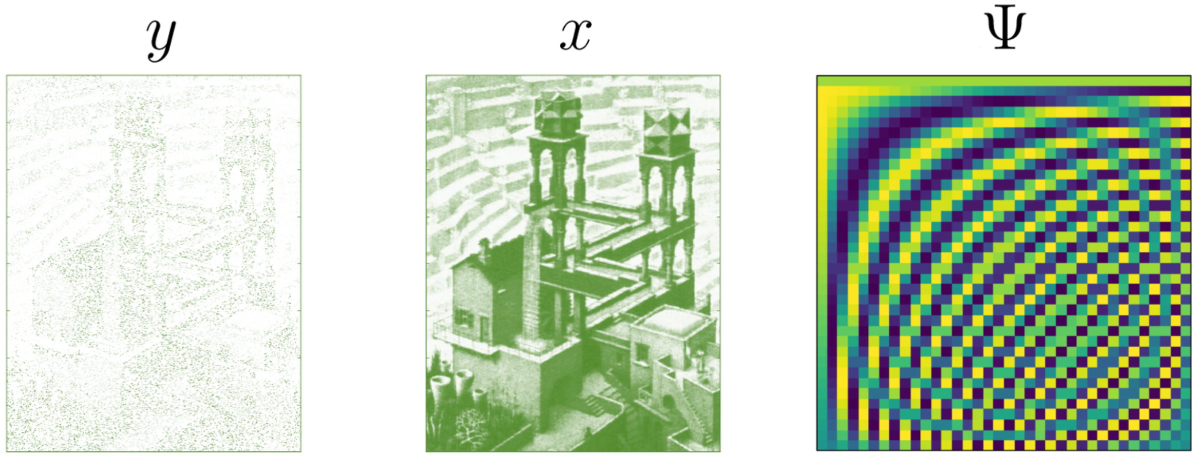


Figure 2.4: Compressed sensing reconstruction of a 2-dimensional image from a sparse set of randomly-sampled pixels using the discrete cosine transform basis functions. Randomly sampled pixels (left) are recovered into the full image (middle) using bases such as the one pictured (right), as adapted from [65].

CHAPTER 3
DEEP LEARNING CLASSIFICATION OF MOTHERBOARD COMPONENTS BY LEVERAGING EM SIDE-CHANNEL SIGNALS

3.1 Overview

As introduced in Chapter Chapter 1, there is no established non-destructive solution to the problem of accurately identifying individual components present on an already-assembled motherboard. While research has shown the ability to identify entire motherboards or large individual components based on their EM signatures [18], [16], to our knowledge there has been no success classifying components already integrated onto a motherboard. This scenario is important for device designers, who need ways of validating that the components on their devices have not been replaced by an untrusted third-party assembler or supplier with counterfeits.

Motivated by these shortcomings, in this work we develop a novel method to solve a variety of component identification tasks using fingerprints from their electromagnetic emanations. Here we train a single Convolutional Neural Network (CNN) architecture to solve several component identification tasks using measurements of their EM emanations in an idle and active state. We show that this approach can distinguish individual IC models within four general classes (processors, memory ICs, power management ICs, and ethernet transceivers). We choose these four component types due to the security threat they pose if counterfeited given their handling of potentially secure data. Additionally, the relative expense of these four compared to other component types like I/O connectors, capacitors, op-amps, etc. makes them more likely targets of counterfeiting. To demonstrate the robustness of our methods we also test our method's capabilities while the IoT motherboards these ICs reside on are operating in "Idle" and "Excited" conditions. This idle state is the most realistic scenario for component validation on an already-assembled device. Then we test on an excited state to demonstrate the performance of our method in a similar setting as [39]. We train our network with a random bootstrap sampling method of cross-validation to achieve more consistently accurate performance over that of a k-Nearest Neighbors baseline. We compare our method against a common off-the-shelf classifier (k-NN) since the problem has no established state-of-the-art solution. It is important to emphasize that we test classification accuracy on held-out devices separate from a training set, rather than subsampling measurements from the same device for training and testing. Additionally, we implement an anomaly detection procedure to show our design can be used to detect components on which the network was never trained. Finally, by examining the signals passing through the neural network, we build hypotheses about the features of different classes of ICs that make them most distinguishable. In doing so, we present a generic approach for accurate and interpretable component-level classification and counterfeit detection on IoT devices.

3.2 Structured signals

When a component on a digital device is supplied power, EM emanations from the component follow consistent patterns based on a host of factors. Among any number of noisy variations in the signal, emanations change as a result of the component's underlying trace architecture and signal activation. With repeated excitation, such as a device clock signal or looping code pattern, we see emanations that are especially noticeable in the frequency domain. In this work, we measure emanations when the device is in an idle state and also when it is activated by a controlled, repeating code execution. This idle state emulates the primary intended use case of this work. Upon receipt of a newly fabricated device, the device designer powers it on to test a variety of functions and confirm the authenticity of its components with non-destructive scans of the EM side channel. We also test with a repeated code execution pattern to emulate the use case of testing the authenticity of a device already deployed and running a known piece of software. Here we discuss these states and the distinguishing features one may look for when understanding the emanated signal.

3.2.1 Component excitations

When powering a device and simply remaining in an idle state, components exhibit minimal structure in their EM emanations besides a peak at the device's clock frequency, as seen in the top-left panel of Figure 3.1. However, with controlled code execution, components can exhibit more structured excitations that are directly related to the activity on the specific integrated circuit. To excite the component, we execute a repeating code pattern to generate amplitude modulation of the looping program activity with the clock frequency [9]. By repeating a pattern of two alternating code sequences (activity X and Y in Figure 3.2), we can control the modulating frequency and duty cycle of amplitude modulation on the device clock frequency. The program execution time spent in one loop versus the other controls duty cycle, while the total execution time of the pattern T_a controls the alternating rate $f_\mathrm{a} = \frac{1}{T_\mathrm{a}}$. That alternating rate shows up as amplitude modulated peaks (excitation) and harmonics on both sides of the centered carrier (clock) frequency, f_c. The result is a frequency domain pattern of spikes at frequencies $f_\mathrm{c} \pm f_\mathrm{a}$ and their harmonics $f_\mathrm{c} \pm 2f_\mathrm{a}, f_\mathrm{c} \pm 3f_\mathrm{a}, \ldots$ with features marked in Figure 3.3.

3.2.2 Features for classification

As seen in each row of Figure 3.1, the distinguishing features of emanations may differ greatly between the idle and excited states. In the idle state, a hand-crafted classifier might extract features related to the clock frequency spread or the trail-off in the sidebands at the edges of the measurement bandwidth. Without color-coding the measurements as in the figure, reliably classifying the signals manually would be difficult. In the excited state, the same features displayed in the idle state are generally present. Additionally, the excited components may be distinguished by features such as the alternating frequency,

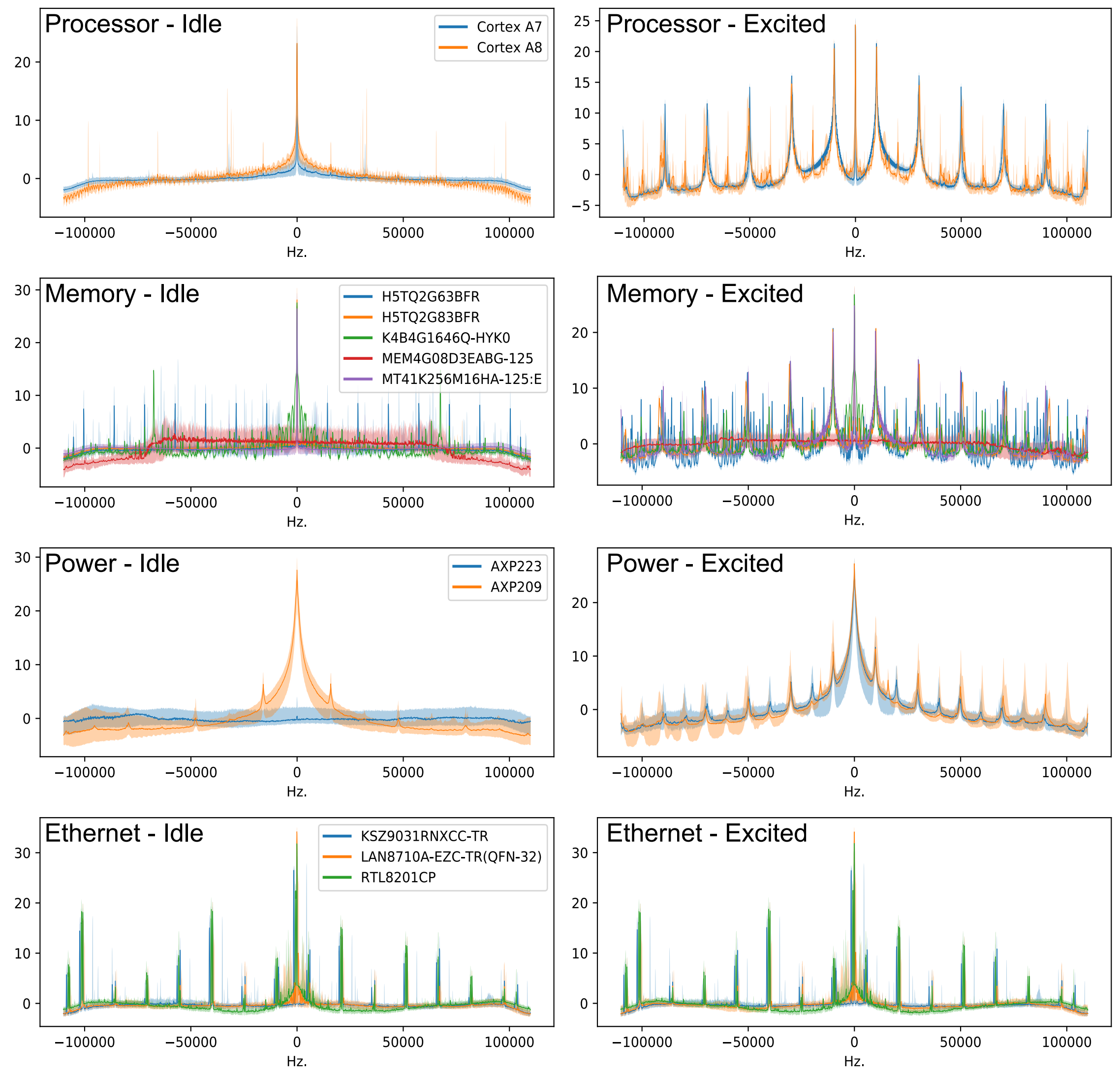


Figure 3.1: EM emanation distribution of each component class and excitation state, horizontally centered by the component carrier frequency and with vertical axis converted to dB and mean subtracted. The median emanation measurement across all components of the same model is drawn with shading below and above corresponding to the 10% and 90% quantiles respectively. Components are measured in the idle (left) and excited (right) states. Both ethernet excitation states look identical but do differ, however imperceptibly.

```
while(true){
  // Execute activity X
  for(i=0; i<x_count; i++){
    ptr1 = (ptr1 & ~mask1) | ((ptr1 + offset) & mask1);
    // X instruction, e.g. a load operation
    value = *ptr1;
  }
  // Execute activity Y
  for(i=0; i<y_count; i++){
    ptr2 = (ptr2 & ~mask2) | ((ptr2 + offset) & mask2);
    // Y instruction, e.g. a store operation
    *ptr2 = value
  }
}
```

Figure 3.2: Pseudo-code generating alternating X/Y excitation [66].

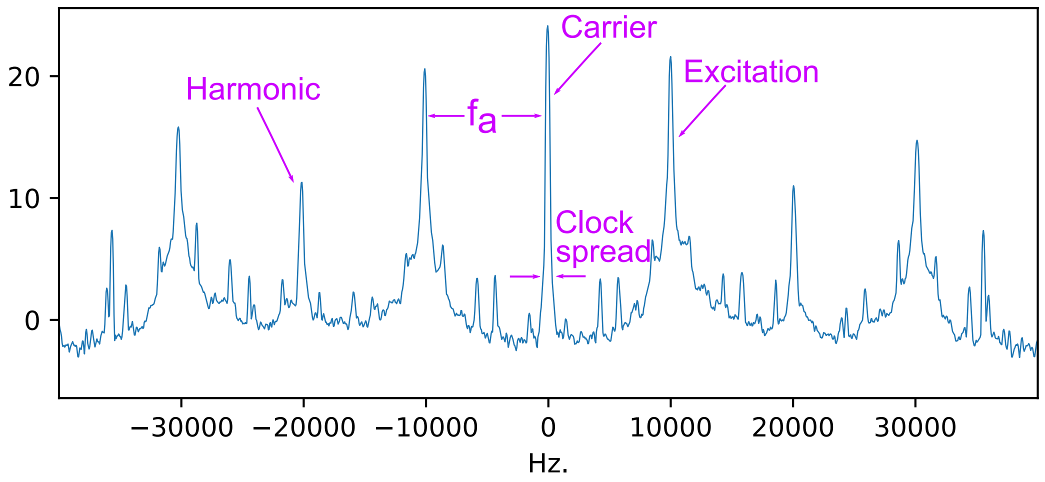


Figure 3.3: Excited EM emanation frequency domain structure in decibel scale, set to zero-mean. The carrier frequency is downconverted to zero and shows a modest amount of spread due to clock frequency variation. Harmonics are present on both sides of the carrier, separated by the alternating frequency f_a of the executed program. Odd harmonics are stronger than even harmonics since the program execution acts similarly to a square wave in the time domain.

relative excitation and harmonic magnitudes, modulations on other unknown sources, and more. While these features are somewhat distinguishable in the idle and excited states when color-coded, it is clear that robust classification of these signals would be difficult by manual inspection. Additionally, given the high dimensionality of the signals, traditional classification models could suffer from the curse of dimensionality; meaning that the noise present throughout the signal overwhelms the ability of a classifier to find the sparse distinguishing features present. To solve these issues and distinguish the components reliably, we develop a preprocessing pipeline and neural network classifier fit for electromagnetic emanation measurements.

3.3 Preprocessing and model architecture

We develop a preprocessing pipeline that is implemented uniformly across eight datasets (four component types with two excitation patterns each) to decrease biases and analyze differences between different datasets. Additionally, we design a convolutional neural network (CNN) classifier to reduce dimensionality while maintaining important discriminatory features of the EM emanations. Here we detail the main preprocessing procedure and our CNN model architecture for classification.

3.3.1 Preprocessing

Each EM emanation measurement can contain millions of samples per second due to the large bandwidth needed to capture the alternating frequency and harmonics of excited components. To reduce noise, reduce the dimensionality of the inputs, and generate several training samples per measurement, we employ the procedure outlined in Algorithm Algorithm 1. EM side-channel signals exhibit significant phase noise, resulting in jittery measurements and a visible spread in the frequencies of the carrier and excitation signals [67]. We designed the preprocessing Algorithm Algorithm 1 to amplify the prominence of frequency spikes and carrier shape above the noise floor, mainly by averaging frequency-domain subsamples captured over time and converting them to decibel scale. This feature selection allows us to generate robust input features to our model from the set of experimental devices available to us.

We subsample each measurement to increase the number of training samples available, reduce the dimensionality of the inputs, and capture potential random time-variability of the signal. From each measurement x with total length N, we subsample non-overlapping windows $z^{(i)}$ of length L. This results in a total of $M = floor(\frac{N}{L})$ segments each corresponding to $T_{\mathrm{S}} = \frac{L}{f_{\mathrm{s}}}$ seconds of the measurement, where f_{s} is the sampling frequency of the measurement device. The resulting measurement matrix $Z \in \mathcal{R}^{M \times L}$ holds each subsample as a row. To reduce the effect of discontinuities at each end of the subsamples, a Hamming window $\mathbf{w}_{\mathrm{h}}$ of length L is applied to each of those subsamples. When measuring EM emanations while executing repeating code patterns, we expect signal structure to be revealed best in the frequency domain, so each subsample is converted to the frequency domain with a Short-Time Fourier Transform (STFT). We transform each complex-valued subsample to its frequency magnitude and ignore phase information. We then perform a small circular shift of the frequency magnitude to center its maximum value which we assume to be the device's clock frequency. Doing so allows our network to focus more on the locations or magnitude of features without requiring potentially difficult training to learn shift-invariant features. Next we convert the signal to decibels to increase the influence of weak signal components like the excitations and their harmonics over the random fluctuations of the noise floor. Non-overlapping groups of length N_G of the M subsamples are averaged to reduce random noise power, resulting in $M_G = ceiling(\frac{M}{N_G})$ subsamples per measurement arranged into a matrix $G \in \mathcal{R}^{M_G \times L}$. Finally, we scale each subsample to have a mean of zero and variance of one to ensure that the model is fed samples with features in a consistent range.

Algorithm 1 Measurement preprocessing algorithm

```
 1: Inputs: Measurement $\mathbf{x} \in \mathcal{R}^N$
 2: Allocate: Subsamples $Z \in \mathcal{R}^{M \times L}$
      Processed samples $G \in \mathcal{R}^{M_G \times L}$
 3: for $i = 1$ to $M$ do
 4:     $Z_{i,*} \leftarrow [\mathbf{x}_{(i \cdot L)}, \ldots, \mathbf{x}_{((i+1) \cdot L)}]$
 5:     $Z_{i,*} \leftarrow Z_{i,*} \odot \mathbf{w}_{\mathrm{h}}$
 6:     $Z_{i,*} \leftarrow STFT(Z_{i,*})$
 7:     $Z_{i,*} \leftarrow \sqrt{\Re(Z_{i,*})^2 + \Im(Z_{i,*})^2}$
 8:     $Z_{i,*} \leftarrow circshift(Z_{i,*})$
 9:     $Z_{i,*} \leftarrow 20 \log_{10}(Z_{i,*})$
10: end for
11: for $j = 1$ to $M_G$ do
12:     $G_{j,*} \leftarrow \frac{1}{N_G} \sum_{n=1}^{N_G} Z_{((j-1) \cdot M_G + n),*}$
13:     $G_{j,*} \leftarrow G_{j,*} - mean(G_{j,*})$
14:     $G_{j,*} \leftarrow \frac{G_{j,*}}{std(G_{j,*})}$
15: end for
```

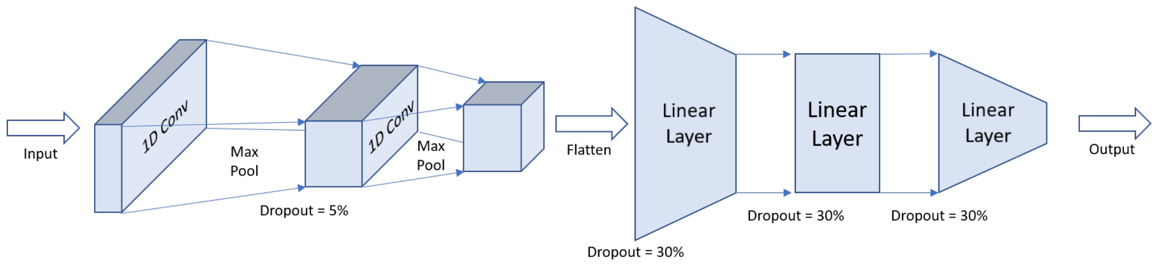


Figure 3.4: High-level schematic of the CNN architecture depicting the convolution blocks (left) and fully-connected blocks (right).

3.3.2 CNN Architecture

While a host of classification algorithms and models exist, few of them are designed to handle high-dimensional inputs with few training samples to learn from, each of which has a sparse set of distinguishing features. Even if other machine learning models prove effective at distinguishing EM emanations when their relevant features are known, CNNs and their variety of trainable and deterministic layers can learn features on-the-fly and allow nearly unlimited model flexibility, all while retaining a level of interpretability. The lightweight convolutional neural network architecture we design here is developed to strike a balance of fast training and inference times while maintaining interpretability and performance on a varied set of component classes and excitations. The architecture is made up of two convolution blocks and three fully-connected blocks as depicted in Figure 3.4 and described next.

Each convolution block uses a one-dimensional convolution with kernel size $K = 3$, stride length $S = 3$, and padding $P = 1$ to significantly downsample their inputs while learning features that are robust to slight jitters of the input frequency spikes. The output of each convolution is fed to a ReLU activation [68] and then a max pooling layer with both kernel size and stride length of four for non-linear representation and more downsampling. The result of the strided convolution and pooling in this block is a downsampling of twelve times in the length of samples fed through the layer. The first convolution block takes as input a univariate subsample of length 4096 and learns three single-channel filters. This multivariate output is passed to the second convolution block with the same basic architecture. This block differs with the addition of a dropout layer [69] randomly zeroing out five percent of its filter weights in order to encourage robustness to sparse noise sources that may be present in the inputs. Additionally, this block differs by learning five separate three-channel filters instead of three separate one-channel filters like the first layer. Between the downsampling due to strided convolutions and pooling in addition to the larger representation power from the increasing numbers of filters, the convolution layers perform a total

Table 3.1: Component models and counts across different device types and component classes. Blank boxes indicate that the device does not have that component. All devices come from the OLinuXino line of development boards and all processors are in the ARM Cortex line. Memory and ethernet chips are abbreviated to the first eight characters for space.

Device Type	Processor	Memory	Power	Ethernet
A10-LIME [70]	A8	K4B4G1646	AXP209	RTL8201C
A13 [71]	A8	H5TQ2G83	AXP209	——
A13-MICRO [72]	A8	H5TQ2G63	——	——
A20-LIME [73]	A7	MT41K256	AXP209	LAN8710A
A20-LIME2 [74]	A7	MT41K256	AXP209	KSZ9031R
A20-MICRO [75]	A7	MT41K256	AXP209	LAN8710A
A33 [76]	A7	MEM4G08D	AXP223	——
Number of Classes (O)	2	5	2	3

effective downsampling of the input feature space of approximately 144-to-5 while only requiring $p_{conv} = 62$ parameters to be learned. The five-channel output of the convolution layers is flattened to a single vector of length 140 to be fed to the linear layers.

The first two fully-connected blocks follow the same structure as each other, differing only in the number of input features per layer. Both blocks learn linear transformations of their input with $M = 50$ neurons, before a dropout layer of 30% and ReLU activation layer. The final linear layer has O neurons, where O is the number of output classes for the specific component classification problem (given in Table Table 3.1). The linear layers require more weights to train than the convolutions; learning $p_{\text{linear}} = 9600 + 51 \cdot O$ parameters. The entire network thus learns $p = p_{\text{conv}} + p_{\text{linear}} = 9662 + 51 \cdot O$ parameters to classify inputs between O classes.

3.4 Experimental design

To measure the ability of our CNN to classify different components on an IoT device, we test an array of different component classes all cross-validated with the same CNN architecture. In this section, we detail the hierarchy of different components tested, the measurements capturing EM emanations from the components, and the training and cross-validation parameters used to obtain the results we present.

3.4.1 Device hierarchy

We use a large set of open source hardware IoT devices manufactured by Olimex to test classification problems for a variety of components. This set of devices spans several copies of each of the $D = 7$ device types: A10-OLinuXino-LIME, A13-OLinuXino, A13-OLinuXino-MICRO, A20-OLinuXino-LIME, A20-OLinuXino-LIME2, A20-OLinuXino-MICRO, and A33-OLinuXino. We classify the component models within each of four classes of components: processors, memory chips, power management ICs, and ethernet transceivers. Though there are seven device types, some of them share the same components, and not all device types have components from each class. The component models on each device type are listed in Table Table 3.1. For each of the four component classes we measure and classify between $O = 2$ and 5 component models.

3.4.2 Cross-validation

An essential piece of developing any machine learning model is a rigorous separation of training, validation, and test samples. As depicted in Figure Figure 3.5, we set aside a group of devices called the training/validation set that are used to select models with the best ability to generalize to unseen data. The remaining devices, called the test set, are used for evaluating performance against a baseline classifier. In our problem scenario, we have several copies each of a variety of different types of IoT devices available to measure. Although the copies each have the same components, measuring multiple copies allows us to train a model that is more robust to manufacturing variabilities of the components or motherboards. To make sure each type of IoT device is represented evenly across the training/validation and test sets, we partition the copies of each device type into groups of $C_{\text{train/val}}$, and C_{test} devices. With D different types of devices, each type with C copies, this

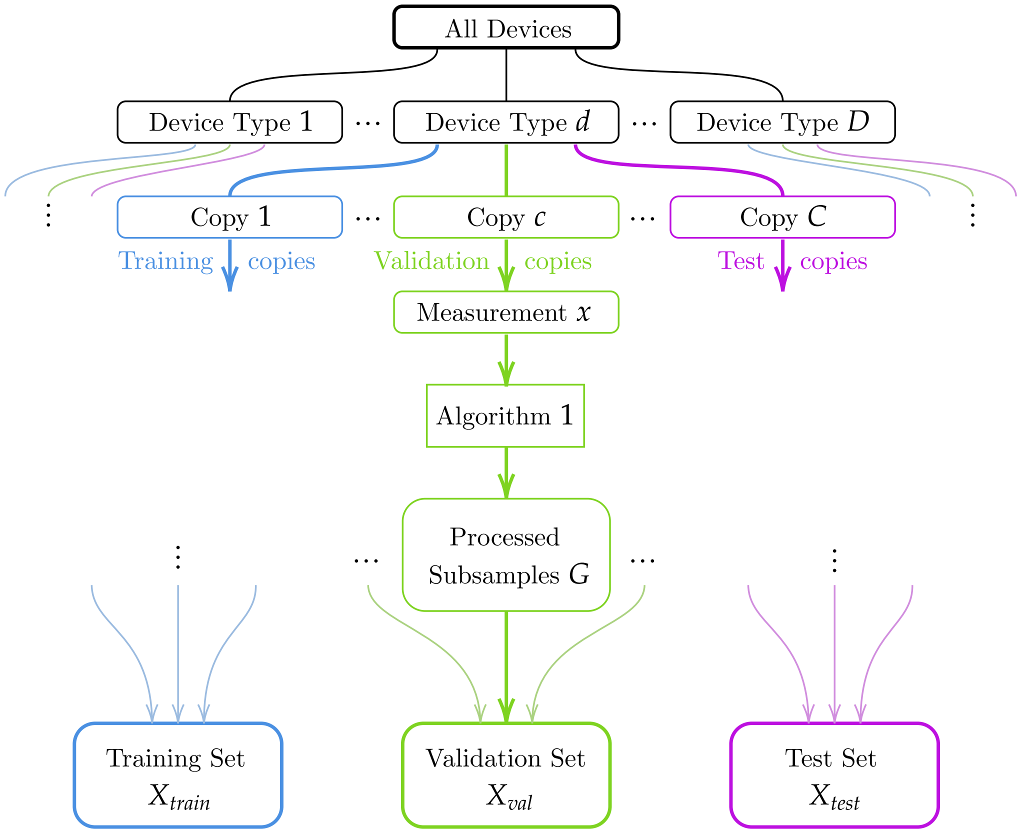


Figure 3.5: Data breakdown and preprocessing pipeline. Device copies are broken into training (blue), validation (green), and test (magenta) groups for preprocessing and finally aggregated together with all other copies of their respective sets.

yields training/validation and test sets of size $D \cdot C_{\text{train/val}}$, and $D \cdot C_{\text{test}}$ respectively.

With just the training/validation set, we use a random bootstrap sampling method to settle on the best-generalized model and the training devices that yielded it. The following procedure is repeated R times to cross-validate the model training. For each device type, we randomly separate the $C_{\text{train/val}}$ device copies into C_{train} training and C_{val} validation devices. Some devices have multiple of the same components (specifically, multiple of the same memory chips), which we group together on each device to maintain consistent set split sizes. The measurements of each device's components are preprocessed into subsamples as detailed in Algorithm Algorithm 1. Then, the $D \cdot C_{\text{train}} \cdot M_G$ subsampled measurement matrices across all training devices are aggregated together into the matrix $X_{\text{train}} \in \mathcal{R}^{(D \cdot C_{\text{train}} \cdot M_G) \times L}$. We then train the supervised model on the data in X_{train} with its corresponding component labels $\mathbf{y}_{\text{train}} \in \mathcal{R}^{(D \cdot C_{\text{train}} \cdot M_G)}$, and evaluate the model performance on a similarly aggregated validation set $X_{\text{val}} \in \mathcal{R}^{(D \cdot C_{\text{val}} \cdot M_G) \times L}$ and its labels. For the purpose of comparing model performance, we use accuracy defined as the proportion of correctly classified samples (True Positives + True Negatives) out of all samples. The model that achieves the best validation accuracy across the R different random training/validation combinations is saved and evaluated on the test set to measure its overall performance.

The value of R can be chosen to strike a balance between computation time and finding the model's best possible validation performance. Given $D \cdot C_{\text{train/val}}$ devices partitioned in this manner, the number of possible combinations of training and validation devices is given in Equation 3.1.

$$\text{combinations} = \binom{C_{\text{train/val}}}{C_{\text{train}}}^{D} \tag{3.1}$$

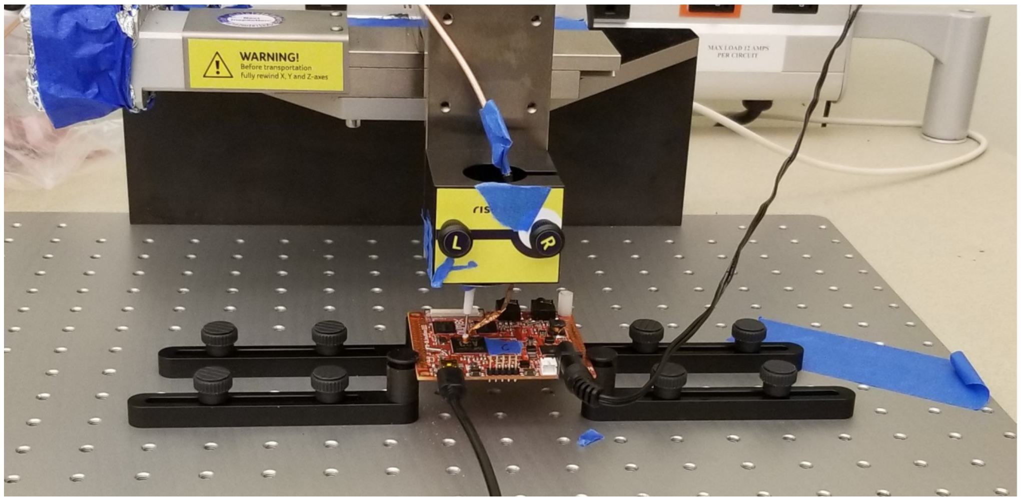


Figure 3.6: Measurement setup for capturing EM emanations from components on motherboards.

3.4.3 Measurements

To capture the EM emanations from each of the components, we measure with the probe setup depicted in Figure Figure 3.6. A handmade circular coil probe with 1 mm spot radius [58] is fixed in place just above the component with the motorized EM Probe Station from Riscure [77]. The probe connects to the Keysight M9391A PXIe Vector Signal Analyzer to record the signal in an In-Phase/Quadrature representation [78]. We isolate signals coming from the processor, memory, and other components by taking advantage of the fact that the different classes of components have different clock frequencies. Knowing the general range of the clock frequency for each component type allows us to isolate them from others by measuring the carrier and any signals modulated onto them in the expected frequency range.

Each component is measured separately while the IoT device is in an idle or excited state as detailed in Section subsection 3.2.1. The excited state consists of a code alternating between addition and load operations at 10 kHz and 50% duty cycle. Each measurement is captured for approximately ten seconds at a sampling rate of 281,600 samples per second for a total measurement length of 2,816,027 samples. Measurements are band limited to a 220 kHz range surrounding the device's carrier frequency as determined by [79] or manually. Each measurement is preprocessed according to Algorithm Algorithm 1 using subsequences of length $L = 4096$ and groupings of $N_G = 50$ subsamples. This results in each ten-second measurement being converted into fourteen subsamples that each represent up to 0.73 seconds of the measurement.

3.4.4 Anomaly Detection

We also test the ability of our model to flag counterfeit IC models on which the network was not trained through an anomaly detection procedure. We perform these tests on memory components since we have the greatest diversity of memory components across device types and training our CNN requires at least two classes to build class-discriminating features. For each of our five tests, we train the CNN with one memory component model held out of the training and validation sets. This component serves as the "unknown" component. The CNN is then trained with only four output neurons now representing the "known" four memory component models. After using the same cross-validation setup as presented in Section subsection 3.4.2, we reintroduce the unknown component model to the testing set along with examples of the four known models. We pass the side channel emanations of the test set consisting of emanations from all five models through our network and use the output activations as the input to the anomaly detection procedure detailed next and similar to those presented in [80] and [81]. We then perform the same test holding out each of the remaining four memory component models from training in turn.

Intuitively, a known component's emanations will have strong correlation with one of the trained components' emanations and consequently produce a strong activation at its corresponding output neuron. A component on which the model was not trained should produce weaker activations across the output neurons due to its low correlation with the emanations of the trained components. We measure the strength of the maximum class output activation relative to the other output neurons as a metric for the CNN's classification confidence. We expect this difference to be large for known components that the CNN classifies confidently, and small for unknown components that have EM emanations unlike any of the known components on which the CNN was trained.

We train a simple generative classifier for anomaly detection with the statistics (μ_{conf}, σ_{conf}) of the CNN's classification confidence for the training samples. We define the classification confidence for a particular sample as the average difference between the maximally activated output neuron and the remaining output neuron activations. Test samples with classification confidence below $\mu_{conf} - \sigma_{conf} F^{-1}(p)$ are classified as unknown, where $F^{-1}(p)$ is the quantile function of a standard gaussian distribution and $p \in [0, 1]$ is the chosen quantile. The choice of p can be made as a trade-off between the rate of components being flagged as potential counterfeits versus being classified as the most similar component model on which the CNN was trained. We choose $p = 0.99$ in our experiments.

3.4.5 Training parameters

We train our CNN in PyTorch [82] using the adapted ADAM optimizer with adjusted weight decay from [83] and default learning rate and weight decay parameters. Weights are randomly initialized by default in PyTorch from a uniform distribution scaled based on the layer type and size. We train with random batches of 50 subsamples without replacement and a maximum of 500 epochs. We implement an early-stopping procedure to reduce training time for iterations that do not converge or show poor generalization performance. To do so, we compute the validation loss every 10 batches and halt training if the loss does not improve after 500 batches. This training procedure yields training times for a single component, excitation type, and cross-validation fold of between 3 and 20 seconds, or more than 18 hours of total training and cross-validation time on our machine with an Intel i7-9800X CPU with 64GB memory and NVIDIA Quadro P400 GPU with 2GB memory.

For each classification problem, we perform the random bootstrapping method outlined in Section subsection 3.4.2 with $R = 50$ training sequences and random training and validation sets for each device type. We have nine or ten copies of each device type, resulting in a total of 273 distinct components measured across the processor, memory, power, and ethernet models with which we employ a 10% / 30% / 60% training/validation/testing set split. We use groups of $C_{\text{train/val}} = 4$ device copies for each type. We train on $C_{\text{train}} = 1$ randomly selected device then validate the model's performance on the remaining $C_{\text{val}} = 3$ devices. The restricted set of training devices and much larger set of testing devices allows us to demonstrate the robustness of our method, while noting that it could be improved with larger sets of devices. From Equation 3.1, we compute that there are between 256 and 16,384 possible sampling combinations depending on the number of device types with that component class. With this many possible combinations, we achieve significant computation savings in the training process by choosing the best validated model across only $R = 50$ random combinations. Then, for each of the eight component classification problems (four component classes with two types of excitation each) we repeat the entire classification pipeline ten times to determine how consistently the CNN performs with so many random factors.

Finally, we train a k-NN classifier with the same preprocessing and cross-validation pipeline to use as a baseline for accuracy comparison. To select the best value of k, for each of the R random bootstrap samples, we train and choose the best performing model across $k = \{1, \ldots, 10\}$ predicted classes.

3.5 Results and Interpretability

We present the results of our classification experiments based on ten trials each of four component classes in two excitation modes, for a total of eighty classification trials or 4000 rounds of training for cross-validation.

Table 3.2: Number of component measurements across all device types in the training, validation, and test sets for each component class. These constitute an approximate 10%, 30%, 60% split for the training, validation, and test sets respectively with a total of 273 distinct components.

Number of Measurements	Processor	Memory	Power	Ethernet
Training Set	7	11	6	4
Validation Set	21	33	18	12
Test Set	40	63	35	23
Total	68	107	59	39

3.5.1 Classification results

We compute the prediction accuracy on the five or six test set copies available for each of the device types. As stated in Section subsection 3.3.1, each measurement is preprocessed into a set of fourteen subsamples which are used to generate a majority-rule prediction for each component measurement. Prediction accuracies here correspond to the exact fraction of components predicted correctly in the test set for each component class. The size of these test sets in each scenario are shown in Table Table 3.2. This number of components varies for each component class due to some device types not having a certain component class (namely, power and ethernet) and other device types having multiples of individual components (memory chips) on a single board. As an example, a memory classification accuracy of 0.952 corresponds to 60 of the 63 test set components being predicted correctly and three predicted incorrectly. A very similar accuracy of 0.957 for ethernet transceivers corresponds to 22 of 23 components predicted correctly and that only one was predicted incorrectly.

Table Table 3.3 shows that our CNN achieves equal or better median accuracy over the baseline k-NN classifier in every test scenario. As shown in Figure Figure 3.7, our CNN architecture achieves a median classification accuracy over 95% in all cases except when classifying power management ICs in the excited mode. In all but that same case, the excited mode produces equal or better results respectively for each of the minimum, median, and maximum accuracies obtained with the idle state measurements.

Table 3.3: Absolute difference of median CNN prediction accuracy over k-NN accuracy across all component classes and excitation states. Positive values mean the CNN outperformed the k-NN baseline.

	Idle	Excited
Processor	0.000	0.025
Memory	0.175	0.000
Power	0.086	0.029
Ethernet	0.043	0.087

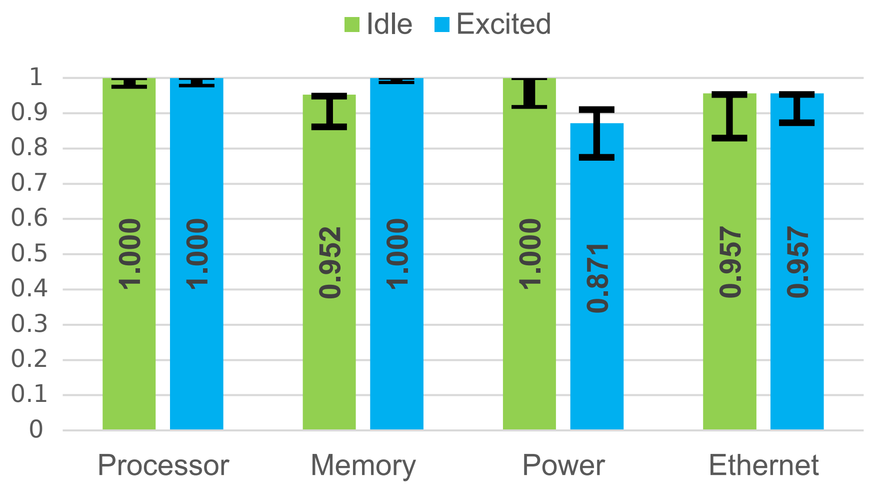


Figure 3.7: Classification accuracy distributions across all four component classes in two excitation states each. Bar heights and inlaid number represent the median prediction accuracy across 10 trials, with bottom and top black markings representing the minimum and maximum accuracies respectively.

As shown in Figure Figure 3.7, the processor model is most consistently predicted correctly, followed by the memory, ethernet, then power ICs. These results follow intuition, since EM emanations related to program activity should be most present near the CPU. Even without these excitations, processor classification results in idle state are still nearly perfect due to the subtle but consistent spread of sideband activity, seen in the top left of Figure Figure 3.1.

The memory chips nearby the processor are spuriously active in the idle state, but consistently active for most classes when the processor is actively transferring data in the excited state. This discrepancy could explain the difference between the two excitation modes' prediction accuracy. We note that the MEM4G08D3EABG-125 memory chip seems to use spread frequency clocking, which results in no visible carrier frequency or modulated excitation in the excited state.

Since both the idle and excited state emanations of ethernet components look nearly identical as we see in Figure Figure 3.1, we expect the network to perform similarly on both. Indeed, we find small differences in the minimum prediction accuracies across the ten trials which we may attribute to the random cross-validation sampling; though overall they share the same median accuracy.

We find that prediction performance suffers when predicting power management ICs in the excited state due to the virtually identical emanations. Since we only had one device type representing the AXP223 power management IC, versus five types with the AXP209, the large class imbalance proved difficult to overcome for the model. There exist many techniques to deal with class imbalances, such as class-weighted sampling or loss functions [84], but we did not test them here so that all processing schema remain the same. The distinct emanation signatures of the different power management ICs in the idle state yield much better prediction accuracy when their emanation signatures differ significantly.

As displayed in Table Table 3.4, we find the highest False Positive or False Negative rates for component models with the fewest overall measurements taken, especially when

Table 3.4: Classification metrics averaged across 10 trials for each component model and excitation type. Shown is the number of ICs of each component model in the test set, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR) for both excitation states. Component types are group by row and model names are abbreviated where necessary.

		Idle				Excited			
Model	# ICs	TPR	TNR	FPR	FNR	TPR	TNR	FPR	FNR
A7	23	1.00	1.00	0.00	0.00	1.00	0.97	0.04	0.00
A8	17	1.00	1.00	0.00	0.00	0.97	1.00	0.00	0.04
H5TQ2G63	5	0.82	0.99	0.01	0.18	0.98	1.00	0.00	0.02
H5TQ2G83	12	0.93	0.97	0.03	0.07	1.00	1.00	0.00	0.00
K4B4G164	6	1.00	0.99	0.01	0.00	1.00	1.00	0.00	0.00
MEM4G08	12	0.93	0.99	0.01	0.07	1.00	1.00	0.00	0.00
MT41K256	28	0.92	0.95	0.05	0.08	1.00	1.00	0.00	0.00
AXP223	6	0.77	1.00	0.00	0.23	0.40	0.96	0.04	0.60
AXP209	29	1.00	0.77	0.23	0.00	0.96	0.40	0.60	0.04
KSZ9031R	5	0.80	1.00	0.00	0.20	0.80	0.99	0.01	0.20
LAN8710A	12	0.99	0.85	0.16	0.01	0.96	0.86	0.15	0.04
RTL8201C	6	0.88	0.99	0.01	0.12	0.90	0.98	0.02	0.10

they exhibit virtually identical spectra to another model. We note that these error rates are most pronounced when we were only able to train on a single IC of a certain model, while other models were common enough across multiple device types to allow several measurements. This was most pronounced for the H5TQ2G63BFR memory IC, AXP223 power IC, and KSZ9031RNXCC-TR and RTL8201CP Ethernet ICs which each had only one training example in our cross-validation setup.

Our anomaly detection results presented in Table Table 3.5 show that our method is able to detect unknown ICs on which the CNN was never trained in many cases. Each group of five rows depicts the results of a test with one memory component held out from training and treated as unknown. Our method yields the lowest accuracy when detecting an anomalous memory IC model which has EM emanations similar to a model with which the CNN was trained. When the CNN is trained on all models directly, it learns feature embeddings that adequately distinguish each IC model. Without training on all IC models, the network relies on feature embeddings whose activations may not differentiate all unknown ICs from known ICs.

3.5.2 Interpreting results

Although we did not extract hand-crafted features for use by our model, a careful inspection of the inputs as they pass through the network can give an indication of what our CNN deemed important for component classification. Figure 3.8 shows how the distribution of training subsamples for the processors in idle state look as they enter the network (top) and exit the convolution layers (bottom). Although the the inputs for each class look similar, we note subtle differences in the height of the carrier peak, presence of small modulated excitations, slope of the sidebands, and sharpness of the trail-off at the outer edges of the measured bandwidth. After passing through the convolution layers of the network, we see that the five filter channels extracted these distinct features from the input. To the naked eye, these outputs can be hypothesized to represent the sideband slopes (blue), carrier strength and sideband slopes (orange), carrier versus excitation peak differences (green), overall sideband decay shape (red), and sideband trail-off (purple). The network learns to extract and emphasize many of the same subtle differences between the inputs that we see with the naked eye.

To add evidence to our interpretation hypotheses, we pass the inputs one layer further in the network to analyze how the convolution layer outputs shown in Figure Figure 3.8 are modified by the first linear layer in what we call its feature activation mapping. As seen in the top of Figure 3.9, each of the 50 neurons' weights (each row of $W \in \mathcal{R}^{50\times140}$) show distinct differences in weight magnitude corresponding to the channels of the convolution layers' output. To understand how these weights affect their inputs, we analyze how the weights of an "average neuron" $\mathbf{w}_{\text{ave}} = \sum_{i=1}^{50} W_{i,*}$ align with the input. We compute the element-wise weighting of the convolution layer outputs with the average neuron and overlay it with the same convolution layer outputs from Figure 3.8. We note in the

Table 3.5: Classification metrics for anomaly detection averaged across 10 trials for memory ICs in an excited state. Each group of five rows represents an individual memory IC model being removed from training and treated as unknown. Shown are the True/False Positive/Negative rates and per-model classification accuracy.

Models	TPR	TNR	FPR	FNR	Accuracy
Unknown	0.980	0.974	0.026	0.020	0.975
H5TQ2G83BFR	1.000	1.000	0.000	0.000	1.000
K4B4G1646Q-HYK0	0.967	1.000	0.000	0.033	0.997
MEM4G08D3EABG-125	0.908	0.998	0.002	0.092	0.981
MT41K256M16HA-125:E	0.993	1.000	0.000	0.007	0.997
H5TQ2G63BFR	0.780	1.000	0.000	0.220	0.983
Unknown	0.492	0.951	0.049	0.508	0.863
K4B4G1646Q-HYK0	1.000	1.000	0.000	0.000	1.000
MEM4G08D3EABG-125	0.883	1.000	0.000	0.117	0.978
MT41K256M16HA-125:E	1.000	0.826	0.174	0.000	0.903
H5TQ2G63BFR	0.840	1.000	0.000	0.160	0.987
H5TQ2G83BFR	1.000	1.000	0.000	0.000	1.000
Unknown	0.350	0.958	0.042	0.650	0.900
MEM4G08D3EABG-125	0.867	1.000	0.000	0.133	0.975
MT41K256M16HA-125:E	1.000	0.889	0.111	0.000	0.938
H5TQ2G63BFR	0.820	1.000	0.000	0.180	0.986
H5TQ2G83BFR	0.983	1.000	0.000	0.017	0.997
K4B4G1646Q-HYK0	1.000	1.000	0.000	0.000	1.000
Unknown	1.000	0.973	0.027	0.000	0.978
MT41K256M16HA-125:E	0.989	1.000	0.000	0.011	0.995
H5TQ2G63BFR	0.920	1.000	0.000	0.080	0.994
H5TQ2G83BFR	1.000	0.876	0.124	0.000	0.900
K4B4G1646Q-HYK0	1.000	1.000	0.000	0.000	1.000
MEM4G08D3EABG-125	0.925	1.000	0.000	0.075	0.986
Unknown	0.775	0.963	0.037	0.225	0.879

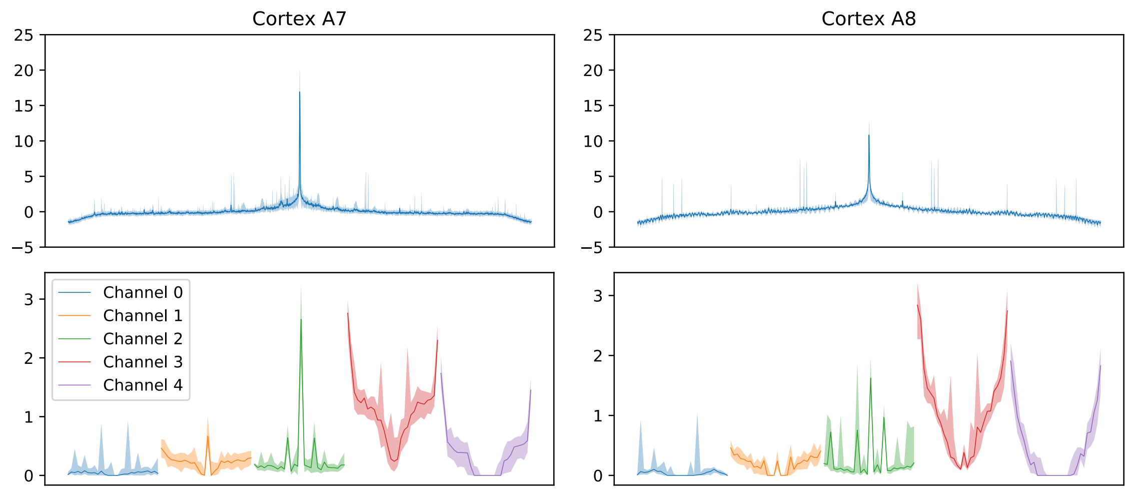


Figure 3.8: Median of training set subsamples for processors in the idle state, scaled to dB and subtracted to zero-mean for each component model with shading below and above corresponding to the 10% and 90% quantiles respectively (top). The flattened output of the same median and shaded quantiles after passing through all convolution layers, just before entering linear layers (bottom).

figures where $\mathbf{w}_{\text{ave}}$ has strong correlation with the input of each class. Without looking at the individual tendencies of each neuron in this layer, we argue that this average neuron highlights which part of its input that *most* of the 50 neurons emphasize to discriminate between classes. These peaks correspond to the carrier frequency peak magnitudes and the outer edges of the sideband trail-off. Nuanced features like sideband slope or carrier versus excitation peak differences may be differentiated by individual neurons, but the average neuron seems to look at these highlighted features as the most important general features.

Although this analysis strays from theoretical bases and likely suffers from some level of hindsight bias, it is worth noting that a similar analysis is entirely impossible with many common classifiers such as the k-NN classifier. When we have a decent idea of important discriminative features from prior work or visual inspection of the input signals, it is useful to compare our intuitions with the features extracted by such a general model. In cases where features are not visually discriminative, this model-guided analysis can be useful to yield insights about the data.

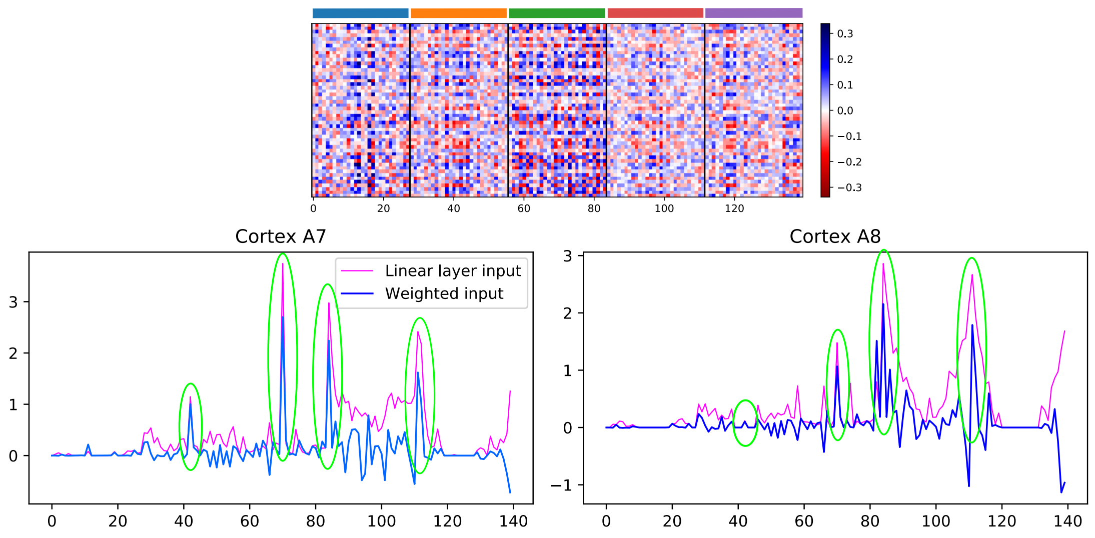


Figure 3.9: First linear layer weights W with black vertical lines separating the weights that correspond to the convolution output channels as color-coded in Figure Figure 3.8 (top). Comparison of the inputs of each component model weighted by the sum of neuron weights (blue), overlaying the inputs (magenta) to the first linear layer (bottom). Weighted inputs with strong correlation (or anti-correlation) to inputs circled in green.

3.6 Conclusions

This work, published in [34], develops a lightweight CNN to classify a broad range of motherboard components in idle and excited states based on their EM emanations. Our robust preprocessing and cross-validation pipeline allows us to train models with strong classification accuracy across a range of component classes. We outperform a k-Nearest Neighbors classifier baseline in all tested cases. We show that a simple anomaly detection procedure can be used with our CNN to detect counterfeit ICs on which the CNN was not trained. While our network performs well across a variety of component classification problems, there is room for improvement with future work. We use a simple anomaly detection procedure based solely off of the CNN's output activations, whereas the state-of-the-art in that domain takes cues from all layers in the CNN. Although our network was intentionally designed for these types of signals, we do not claim to have made a comprehensive architecture search and it is likely that architecture tweaks could improve performance further. This is especially likely to be the case with individual models tuned to the traits of each of the eight individual component identification problems. To emphasize the utility of this end-to-end trained method, we provide an in-depth analysis of signals as they pass through our model. This process allows us to understand the discriminative features of different components when they may not be visually apparent. Our results demonstrate that a lightweight CNN can classify the diverse EM emanations accurately while generating insights about their discriminative features for individual integrated circuit components on an assembled motherboard.

CHAPTER 4
FEATURE SELECTION FOR NON-DESTRUCTIVE DETECTION OF HARDWARE TROJANS USING HYPERSPECTRAL SCANNING

4.1 Overview

While large-scale differences between integrated circuits can be detected through the electromagnetic side channel, small intrusions like hardware Trojans are much more difficult to detect; especially when dormant. Prior work has demonstrated the ability to detect hardware trojans either after activation [85], [4] or only recently for large dormant Trojans [25]. For Trojans designed to be stealthy, minimizing circuit resource usage compared to the rest of a circuit and only activating under rare circumstances can help avoid detection in most routine functional testing scenarios [19]. While destructive testing methods can potentially detect these small dormant Trojans, testing a significant portion of manufactured devices would require significant time and effort and necessarily lead to significant losses by destroying chips. Thus, it is crucial to develop non-destructive methods that can be used to validate hardware security at high throughput before the devices are deployed.

To overcome the shortcomings of existing non-destructive HT detection methods and approach the reliable detection performance of destructive methods, we develop a novel approach that detects dormant HTs by informed sampling of a hyperspectral backscattering EM side-channel measurement space. The impedance-based backscattering EM side channel has shown promise for detecting dormant HTs when scanned at a single chip location, so we take advantage of spatial variation of the side channel to capture more information that may expose a covert Trojan. Using a high-resolution probe, we spatially scan the IC across space and frequency to generate point-scanned hyperspectral images of the backscattered EM signal. We develop novel filtering and active learning techniques to sift through the large hyperspectral feature space to selectively capture the most reliable measurements that improve HT detection robustness while significantly reducing scanning overhead at test time. We evaluate our new techniques with three benchmark circuits featuring distinct Trojan designs and validate across multiple hardware instances to demonstrate the robustness of our novel measurement and analysis techniques. We show that our selective scanning method is used to detect dormant hardware Trojans as small as 0.03% of the circuit; which is 14 times smaller than prior work. This point-scanning hyperspectral measurement methodology shows promise for future efforts improving detection accuracy and speed.

4.2 Dormant Hardware Trojan Detection

4.2.1 Hardware Trojan Threat Model

Hardware Trojans present on a device represent a significantly more serious risk than simple design errors, natural runtime errors, or random bit-flip errors. While random errors may be accounted for with error correction code, HTs act covertly and adversarially which can have unforeseen effects on otherwise secure software. There has been much work to understand the threat that HTs pose [40], [21], [41] while also further understanding and characterizing their implementation [42], [43], [44], [45]. A device designer cannot trust the origins of its devices blindly due to the ever-increasing number of entities involved in their development and manufacture. A fabricated IC is vulnerable to potentially malicious actors across the design, fabrication, testing, packaging, and other stages. Since all of these steps are typically completed by separate specialized entities, perhaps even spread across different countries, ICs fabricated in the most robust supply chains may still be vulnerable to HT injection by bad individual, corporate, or political actors. A typical HT threat might modify the design of an application-specific integrated circuit (ASIC), digital signal processor, microprocessor, or other hardware at an untrusted foundry [20]. Even when a trusted designer securely develops circuit modules and generate their layouts using trusted design tools, sending the layouts to an untrusted foundry for manufacture leaves a window for potential modification of the circuit layouts [46], [20]. Thus, it is necessary to perform final design verification of the functional IC before deploying it to detect the presence of HTs that may be added somewhere in the device manufacturing supply chain.

The recent prevalence of field programmable gate arrays (FPGAs) for prototyping or other specialized critical applications also presents a risk for malicious Trojans inserted as firmware modifications through the FPGA's bitstream [23], [24]. Even if an FPGA's configurable logic blocks and programmable interconnects are not maliciously modified at the hardware level, the addition of covert Trojans into the bitstream can implement malicious logic independent of the hardware platform [24]. Though the performance gap between FPGAs and ASICs is reducing, FPGAs do not necessarily have the same circuit properties as ICs and further investigation of HTs inserted into ASICs is necessary to validate detection techniques [23]. However, the path length and resulting impedance changes caused by an inserted Trojan that our HT detection technique relies on could still be present whether in FPGA or ASIC [23], [41]. The Trojans analyzed in this work are implemented on FPGA platforms versus custom ASICs for reasons of practicality.

The internally-activated, condition-based HTs we investigate in this paper are designed to take up very little circuit resources and be activated under very rare circumstances to avoid detection by common testing methods. This type of HT typically consists of a trigger and normally-dormant payload circuit as seen in Figure 2.2, where the trigger circuit monitors the main circuit for a specific combination or sequence of inputs, state, or other conditions before activating the payload circuit when those conditions are met [19], [20]. The payload circuit draws negligible current while dormant, which is its main defense to avoid detection by traditional HT detection methods. As shown in Figure 4.1, a small trigger circuit makes the entire Trojan incredibly difficult to detect; with statistically significant differences showing up in as little as 0.2% of features across space and frequency. Once activated, the HT initiates its malicious purpose which may be functional or non-functional.

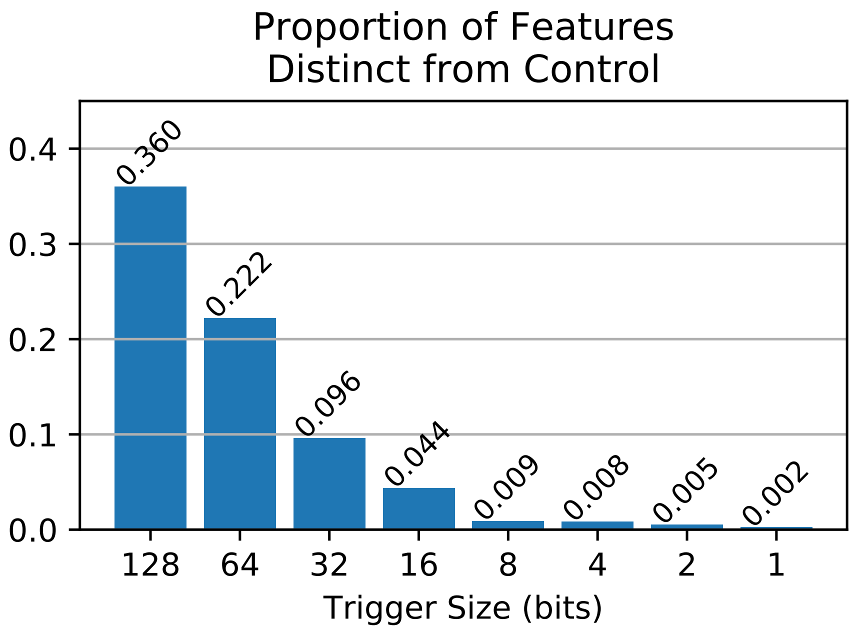


Figure 4.1: Proportion of total hyperspectral features showing statistically significant difference between uninfected and infected AES-T1800 circuit.

A functional HT might change the value of the main circuit's outputs or communicate secure information externally while a non-functional HT may cause rapid battery drain or leak sensitive information through a side channel, among many other possibilities. Since the effects of an activated Trojan can be so devastating, it is critical to detect their presence before activation or, ideally, before the device is even deployed.

4.2.2 Backscattering EM Side Channel

As noted in chapter 2, EM side-channel signals are emissions due to changes in current in the ICs and conductive traces on electronic devices [10], [47]. These emissions tend to show up as sharp spikes with harmonics in the frequency domain due to consistent transistor switching from a clock signal or looping program activity [9]. The authors of [25] developed the backscattering EM side channel to reduce issues with noise and interference with the original EM side channel while improving the ability to detect dormant HTs that cannot be found with current-based side channels. In that work, a 3.031 GHz continuous wave sinusoid is transmitted toward the IC and the reflected (backscattered) signal is measured with a probe. That work draws inspiration from the way that RFID tags can transmit information by switching between two antennas with different impedance [49]. The RFID reader's transmitted signal at frequency f_t acts as a carrier for the impedance of the tag that alternates at f_c, resulting in an amplitude-modulated signal returning bits of information back to the reader. Similarly, the backscattering EM side channel is captured by transmitting a signal which is modulated depending on the state of an IC. The backscattered signal reflects back differently depending on the transistors' states, which have different impedance when connected to a voltage source versus being grounded [50]. The majority of transistor switching occurs at the IC's clock frequency f_c; picked up by a probe as modulated spikes in the frequency domain at $f_t \pm f_c, \pm 2f_c, \ldots$ as shown for the real circuit activity in Figure 2.3. Variations of these spikes occur depending on the specific circuit architecture and the transistor activity in the region of the probe. These variations may flag the presence of a Trojan in the circuit when compared against a reference signal. Circuits infected with dormant condition-based HTs may be detected through the backscattering EM side channel because their distinct transistor architecture reflects EM signals differently than their uninfected circuit counterparts. While we do not test other types of internally-activated HTs here, the addition of any Trojan that modifies the IC layout could potentially affect the backscattered signal enough to be detectable. The backscattering EM side channel has the main advantages over the original EM side channel of being impedance-based to detect inactive Trojans and being adjustable to allow the user to avoid frequency bands with significant interference.

4.2.3 Hyperspectral Imaging

Hyperspectral imaging (HSI) typically refers to the collection of many two-dimensional images at different frequency bands in the visible or near-visible light spectrum [86], [56]. While normal color photos are represented in 3-dimensional space with two spatial dimensions and one frequency dimension for the red, green, and blue channels, HSI captures tens or hundreds of frequencies in a 3-dimensional hypercube of data per image. While visible light HSI is typically used in fields such as geoscience [51] and biomedicine [52], [53], some HSI techniques have been used to analyze the quality and composition of printed circuit boards [54], [55] for electronics recycling. While these traditional visible light HSI methods do not penetrate an IC's packaging, the backscattering EM side channel reflects differently based on the underlying architecture. Prior work spatially scanned the EM side channel to detect large HTs, but did not explicitly target frequency information [87]. Point scanning HSI repeatedly captures the backscattered signal at individual spatio-spectral locations until the entire hyperspectral space is measured [53].

The unique transistor activity for specific ICs and different programs will yield side-channel emanations that vary across the chip area *and* frequency. The authors of [25] used this knowledge to manually position a probe at a single location above an FPGA chip to maximize the SNR of the received signal across a range of frequencies and detect the presence of HTs. While this approach produces carriers and sideband harmonics with good SNR, it is not always true that locations with high SNR correlate to those that best reveal HTs as demonstrated in Subsection 4.2.4. The unique transistor architecture and activity across the chip area will result in different spectral characteristics, as seen in Figure 4.2, so it is important to capture that variability to detect elusive Trojans. Without knowledge of a specific Trojan threat, it is impossible to know where an HT may present itself on the chip.

4.2.4 HT Detection Baselines

To demonstrate the importance of hyperspectral scanning and informed feature selection, we present four baseline HT detection methods. The first method is designed to exemplify why scanning multiple frequencies but at only one chip location as in [25] can be unreliable for detecting Trojans. The second shows that even naïvely scanning the entire IC outperforms single-location scanning. The third baseline demonstrates a straightforward

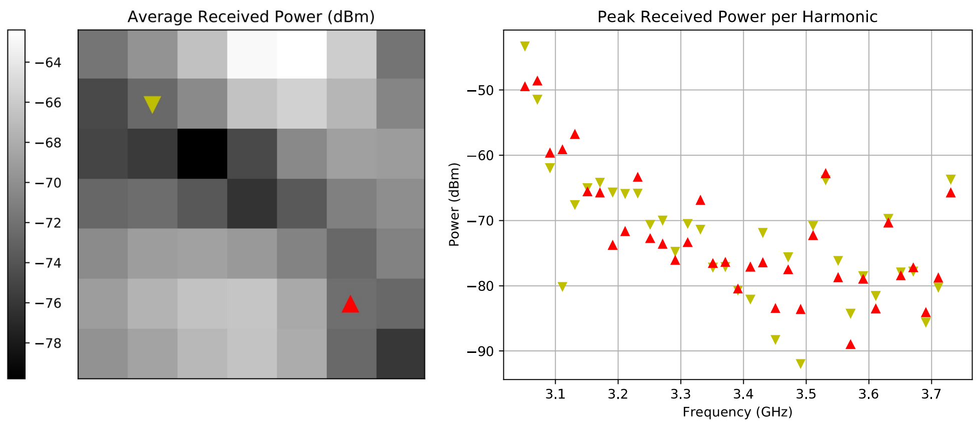


Figure 4.2: Average backscattered EM side-channel power across an FPGA with two points of equal power marked (left). Corresponding power across a 700 MHz span for the same points on the IC, showing unique spectral characteristics (right).

averaging approach to reduce signal noise across the hyperspectral measurement space. And finally, the fourth demonstrates that smaller Trojans may be hidden within the noise of the entire hyperspectral scan, motivating an informed feature selection method. Each baseline is evaluated by clustering real measured data samples between an HT-infected and uninfected AES-T1800 benchmark circuit, further described in Subsection 4.4.1, for a range of Trojan trigger sizes. The performance of each baseline averaged across 100 random trials is presented in Figure 4.3 which demonstrates that Trojans with small triggers are difficult to reliably detect without hyperspectral scanning and feature selection that reduces the full measurement space.

The first baseline ("Max Power" in Figure 4.3) acts as a counterintuitive example to demonstrate why manual probe placement is unreliable without knowing the location of Trojan circuitry on the IC. Here we test the seemingly-reasonable hypothesis that measuring only the location with highest SNR would provide the best ability to distinguish a control and Trojan. To do so, we measure an uninfected device at all hyperspectral locations and select the physical location with the greatest average received power across all harmonic frequencies to measure at test time. Like in prior work using the backscattering side channel [25], we then compare measurements taken across frequency at a single lo-

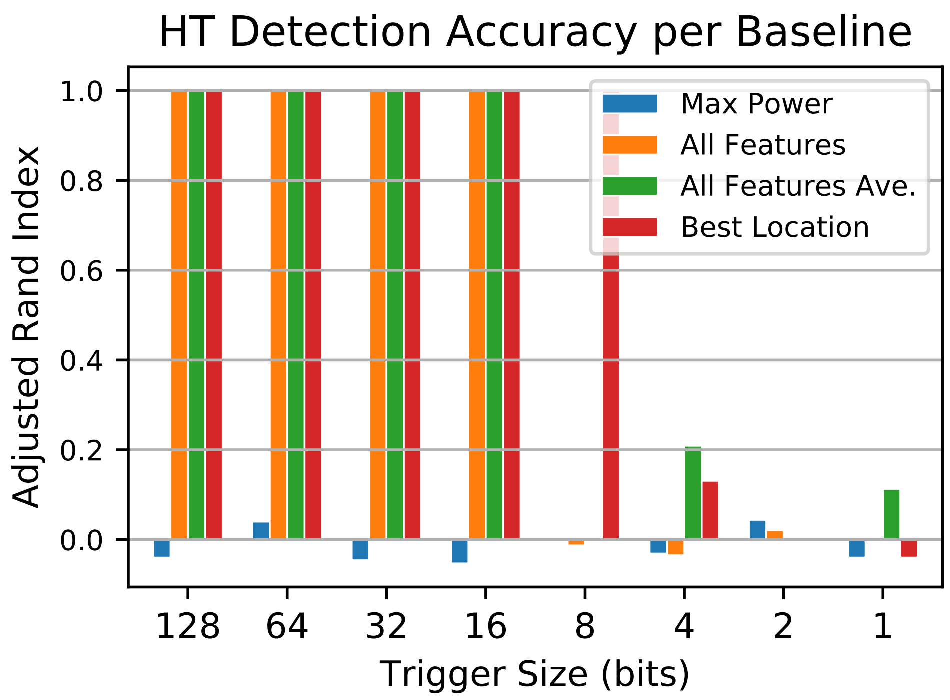


Figure 4.3: Clustering performance (adjusted rand index $\in [-1, 1]$) between an uninfected and infected AES circuit for a range of trigger sizes with the four baseline HT detection methods. Scanning at the maximum power location was uninformative for HT detection, using all hyperspectral features detects HTs smaller than prior work [25], averaging scans provides negligible improvement, and the best possible location can unveil even smaller Trojans.

cation on a test device to those of an uninfected circuit. The poor clustering performance demonstrates clearly that high SNR measurements do not necessarily correlate to locations of good discriminability between an uninfected and infected circuit. Without knowing the location(s) on the IC which will best differentiate an uninfected from an infected circuit, it is necessary to sample across the entire hyperspectral space.

Without knowing where on the chip that Trojans might reveal themselves, the second baseline ("All Features") naïvely uses the features of the entire hyperspectral space to compare test measurements to the control. Simply using all information from these full-chip scans allows us to detect triggers down to 16 bits, whereas prior work scanning one location was only able to detect down to 64 bits without false positives [25]. This improvement shows that hyperspectral scanning is necessary to detect smaller Trojans, but the smallest Trojans are still obscured by the sheer number of features in the hyperspectral space. A simple test shown in Figure 4.4 confirms that the smallest Trojans are undetectable without significantly reducing the feature set. In that test, we remove features in order from least to greatest statistical distinction of the test samples from the uninfected control samples. While not perfectly monotonic, the results there show that some sort of feature selection may be necessary to detect the smallest trojans; even thought hyperspectral scanning already improves detection performance significantly.

The third baseline ("All Features Ave.") averages groups of three full-chip scans to reduce noise power [88]. This method slightly improves the ability to detect some triggers below 16 bits, but the small differences in the backscattered EM side channel caused by these triggers are still masked by the large feature space and noise variation of a full hyperspectral scan. Moreover, the increased sampling cost (tripling the number of scans, in this case) is hardly worth the mild performance increase when sampling is so time-intensive. As demonstrated by the second baseline, it is apparent that the smallest Trojans still remain hidden in the large hyperspectral measurement space.

A final baseline ("Best Location") further motivates a feature selection strategy to re-

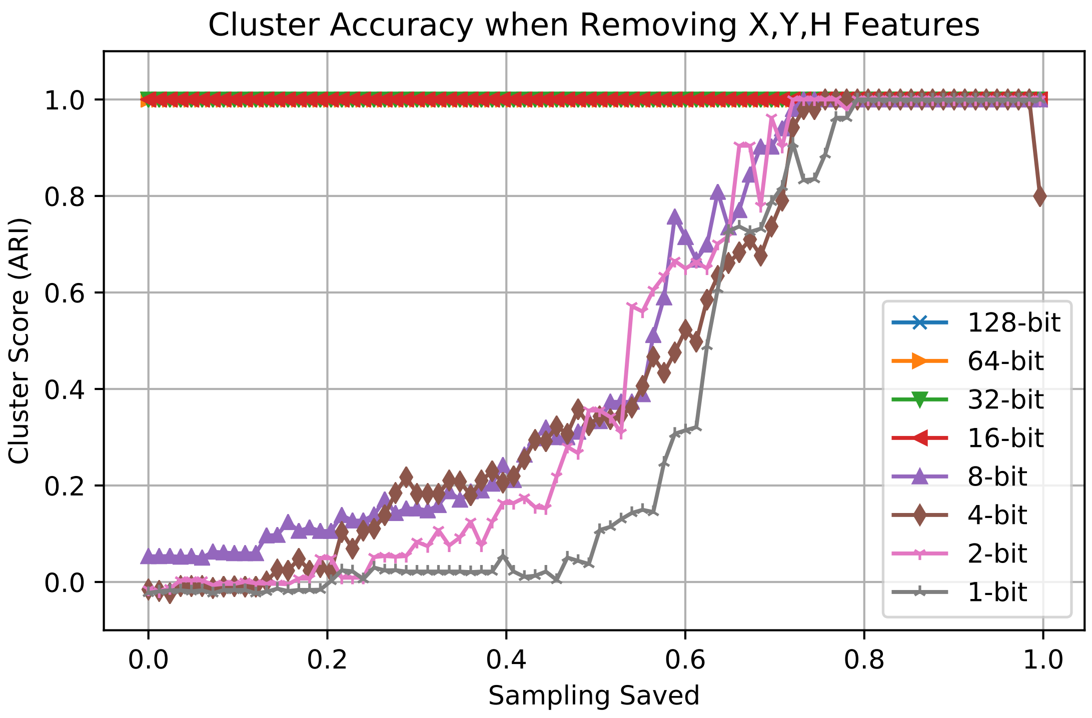


Figure 4.4: HT detection accuracy when clustering as the number of features filtered out from the sampling set increases (resulting in less sampling from left to right), for Trojans with triggers of different sizes on the AES circuit.

duce the ability of noisy data that obscures covert Trojans which are only revealed at a sparse set of hyperspectral points. This baseline emulates the best case scenario where the user, by chance, happens to measure the exact single location on the IC where the harmonics are most different between the control and test sets. While it is impossible to know which location that is without knowing the Trojan and circuit characteristics, this shows better performance than a full-chip scan by being the only baseline presented here to consistently discern the 8-bit trigger. These baselines demonstrate not only the importance of hyperspectral scanning from multiple IC locations, but also the utility of informed feature selection to improve HT detection for the smallest and consequently most stealthy HTs. To that end, we investigate a point scanning approach to find the reduced set of spatio-spectral points that indicate the presence of HTs while avoiding measuring the points that simply contribute irrelevant information or noise.

4.3 A Novel Feature Selection Strategy for Hardware Trojan Detection using Hyperspectral Scanning

To capture the full spatial variation of the backscattering EM side-channel emanations, we might traditionally raster scan the chip area while measuring a range of frequencies at each step to develop hyperspectral images. At each x, y location on an uninfected chip, we measure the peak height of the harmonics that show up due to the amplitude modulation between the transmitted signal and the IC's clock frequency. These harmonics are spaced at frequencies $f_t + hf_c$ with $h \in 1 \dots H$ for the H harmonics measured. Each hyperspectral scan $s \in 1 \dots S$ thus takes the shape $\mathbf{Z}_s \in \mathcal{R}^{X \times Y \times H}$ where X and Y are the number of scan locations in the horizontal and vertical directions. While point-scanning HSI in this scenario allows us complete flexibility of which locations or frequencies to measure, it also heavily increases the measurements, and therefore time, needed to scan a single chip. Additionally, we do not know which hyperspectral features might best differentiate between infected and uninfected circuits, so our HT detection algorithm must take into account all reliable measurements we can capture. The rest of this section details the two parts making up a novel sampling and filtering method to reduce measurement cost while more robustly detecting smaller Trojans. Finally we present our Trojan detection algorithm.

4.3.1 Feature Pre-Filtering

While capturing the backscattered signals across the entire chip surface may uncover more features that highlight certain Trojans, not all these data are useful for detecting each Trojan. The sparse set of distinguishing features for stealthy Trojans may be occluded by the large proportion of features that are irrelevant for this detection task. Ideally, one would only measure the set of features that reliably signal the presence of a Trojan, but this unpredictable set of features will differ based on the type, size, and location of the Trojan. For this reason, we first learn which features are most unreliable or noisy from the training set of uninfected circuit data.

In a different scenario where repeated sampling is fast and cheap, we might use a dimensionality reduction technique like Principal Component Analysis (PCA) or an autoencoder neural network to reduce the feature dimensionality. In our scenario however, data capture is time-intensive and allows many fewer measurements captured than the dimensionality of the feature space. As detailed further in Section 4.4, we capture up to 10 measurements per IC of the entire hyperspectral space which can have more than 1700 features; rendering traditional dimensionality reduction methods unusable. Rather than using these techniques that require extensive data to learn a reduced feature space, we design a feature filtering method that still retains an explicit correspondence between the full and reduced feature sets. Specifically, we rank the features based on their standard deviation across measurements in the training set and filter features out by starting with those having highest deviation. As seen in Figure 4.5, manufacturing variability between ICs accounts for much higher variations than measurement noise alone. For each of the $X \cdot Y \cdot H$ hyperspectral points measured on a single FPGA, interference and measurement noise generally account for less than 1 dB of the measurement's variation. Measuring the same hyperspectral points across a variety of FPGAs shows that manufacturing variability can account for up to 10 dB of variation per point. The large spread in the measurements of these features, even between known uninfected circuits, would mask the small differences that HTs make on the backscattered signal.

As a simple confirmation that this feature filtering is useful, we compute the correlation between random control and test samples for the same benchmark Trojan circuit as in Subsection 4.2.4. With the full data, control samples have a correlation factor of 0.797 with other control samples and 0.785 with test samples from a 4-bit Trojan; a difference of 0.012. With half of the features removed, this difference in correlation increases with control-control correlating at 0.756 and control-test at 0.721; an improved difference of 0.035. While control samples look slightly more different from one another with this feature filtering, the Trojan samples are significantly more distinct than before. Thus, removing those highly-variant features should allow the smaller differences caused by HTs to be detected. At the same time, this allows the use of information learned from modeling a known set of uninfected ICs at training time to then significantly reduce the hyperspectral space needing to be measured at test time.

4.3.2 Active Sampling

While noise or interference are significantly reduced compared to the original EM side channel, these variations in the backscattering EM side channel of a device under test (DUT) still leads to some amount of measurement uncertainty. A common way to reduce noise power of a steady state signal is to average samples captured in windows over time [88] as tested in the "All Features Ave." baseline in Figure 4.3. While this averaging increases the SNR of the steady state signal over noise and any other transient circuit activity, it comes at the cost of heavily increased sampling requirements. Rather than capturing a set number of samples over time at every point in the hyperspectral space, we devise a

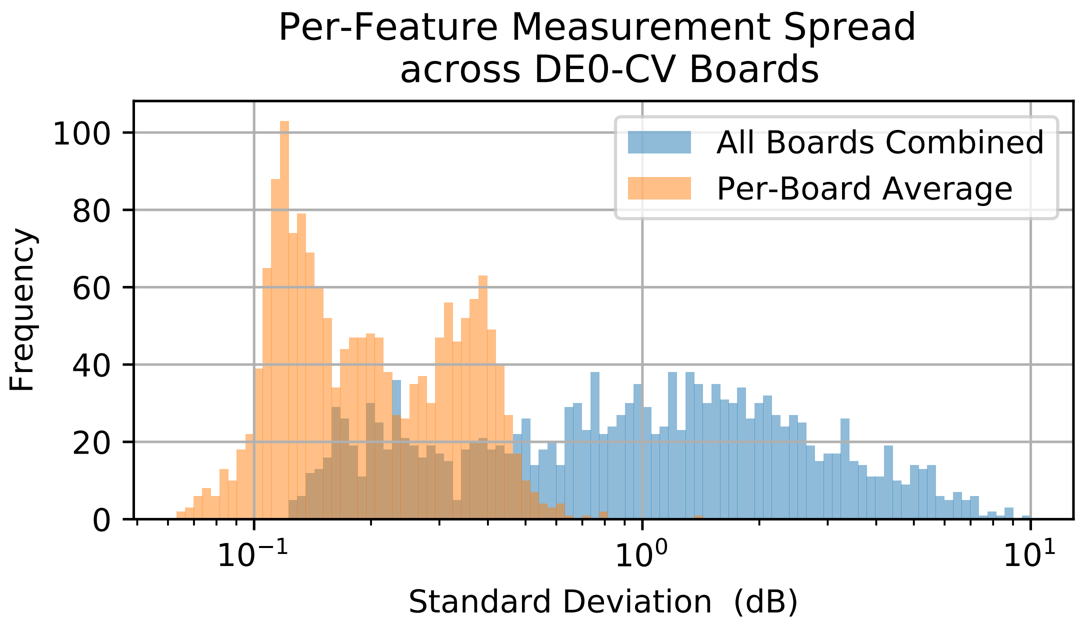


Figure 4.5: Distribution of measured received power for hyperspectral point across all measured FPGA boards (blue) and averaged per board (orange).

sampling method to reduce wasted sampling when measurements exhibit minimal variation. Since the natural variation over time of each feature in the hyperspectral space may be significantly different due to the local differences in trace architecture, we cannot simply set a variation threshold under which we say the measurements are reliable. Instead, we compare the distribution of samples of a given hyperspectral point on the DUT to the distribution of the same point's samples on the control devices.

This idea lends naturally to an active sampling method which can significantly reduce the number of samples we capture at each hyperspectral point (x, y, h). At each point, we have a set of control measurements $\mathbf{Z}_c(:, x, y, h) \in \mathcal{R}^{N \cdot S}$ with some variation due to noise, interference, manufacturing variabilities, etc. Using a statistical test, we can check if samples $\mathbf{Z}_t(:, x, y, h) \in \mathcal{R}^{S_{\text{xyh}}}$ of the DUT match the distribution of those from the control devices. Here we use the two-sample Kolmogorov-Smirnov test in Equation 4.1 measuring the distance between two cumulative distribution functions to evaluate the hypothesis that the test and control distributions are the same. Another test option may be the F test which specifically tests if two distributions have the same mean but requires the assumption that the samples follow a Normal distribution. To avoid making more strict assumptions, we use the KS test here but find negligible differences in performance using other statistical tests. These tests yield a p-value, where p near 0 implies that the distributions are different and p near 1 implies that they are the same. Our active method samples each feature on the DUT until those samples either significantly match or diverge from the control distribution, i.e. $p_{\text{xyh}} < p_{\text{low}} \vee p_{\text{xyh}} > p_{\text{high}}$, given some p-value thresholds p_{low} and p_{high}. Intuitively, this method continues sampling only if we are unsure whether the feature matches the control distribution or not and stops sampling once we can confidently match with or distinguish from the control distribution. This method allows us to spend the most measurement effort on only those features that do not obviously match or diverge from the control distribution. To ensure no feature is sampled indefinitely, we set a maximum sampling budget S per hyperspectral point.

$$D_{KS} = \sup |\mathbf{F}_{\text{control},L} - \mathbf{F}_{\text{test},M}| \tag{4.1}$$

4.3.3 Hyperspectral Dormant HT Detection Algorithm

Finally, we design our novel HT detection algorithm with two main constraints in mind. The first constraint is that we cannot assume which hyperspectral features may best highlight differences between an infected and uninfected circuit. Prior work showed that a Trojan's presence has unique effects on the backscattering EM side channel depending on both the circuit and Trojan's architecture and location [25], so our algorithm cannot be biased toward certain features without *a priori* knowledge of the circuit and potential Trojans. The second constraint is that we may not sample all features an equal number of times. As detailed previously in Subsection 4.3.2, we sample individual hyperspectral features an undetermined number of times until we meet some threshold of confidence about the measurement. Some features may only be sampled once and others may be sampled the maximum S times. These two design considerations lead us to develop a simple metric with which we mark the specific we are measuring on a DUT as infected or not. Given a set of up to S measurements per feature of a known uninfected set of ICs, $\{\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_N\}$, we aggregate them into one set of control measurements $\mathbf{Z}_c \in \mathcal{R}^{(N \cdot S) \times X \times Y \times H}$. The corresponding measurements of the IC on a DUT are labeled $\mathbf{Z}_t \in \mathcal{R}^{S \times X \times Y \times H}$ where the feature at location (x, y, h) is sampled $S_{\text{xyh}} \in [0, S]$ times. We use these data to measure the total distance between the average control and test measurements with the $L1$ norm, as shown in Equation 4.2. A threshold on this distance metric is set based on the distribution of the distance from individual known-uninfected ICs to the average of all other known-uninfected IC measurements. If a given DUT differs from the average control measurements by more than that threshold for a designated confidence level, it is marked as infected with an HT.

$$D(\mathbf{Z}_t, \mathbf{Z}_c) = \sum_{x,y,h} \left| \frac{1}{S_{\text{xyh}}} \sum_{s=1}^{S_{xyh}} \mathbf{Z}_t(s, x, y, h) - \frac{1}{S} \sum_{s=1}^{S} \mathbf{Z}_c(s, x, y, h) \right| \tag{4.2}$$

4.4 Trojan Design and Hyperspectral Measurement Setup

Care is taken throughout the experimental design to ensure that Trojans are tested in as stealthy a scenario as possible while also being completely removed for the uninfected circuit designs. Our measurement setup also ensures consistent data capture to offer as repeatable an implementation as possible. An overview of the experimental process flow is depicted in Figure 4.6.

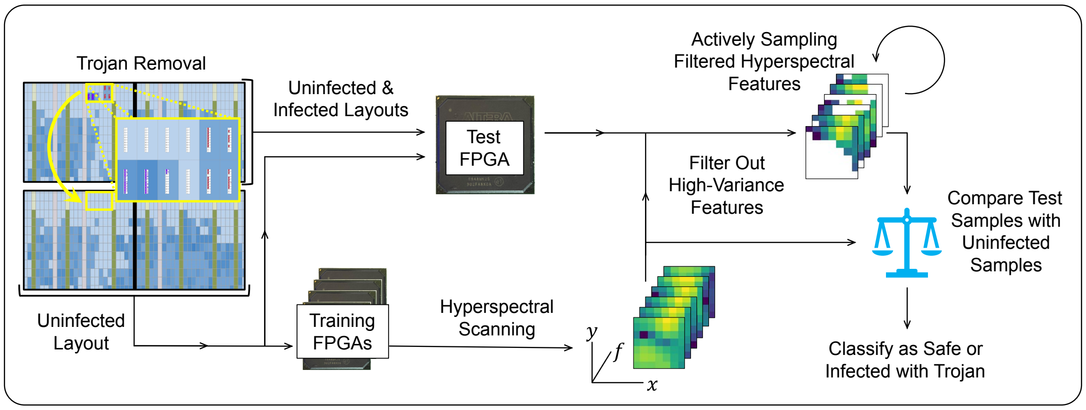

Trojan Removal
Uninfected & Infected Layouts
Test FPGA
Actively Sampling Filtered Hyperspectral Features
Filter Out High-Variance Features
Compare Test Samples with Uninfected Samples
Uninfected Layout
Training FPGAs
Hyperspectral Scanning
y
f
x
Classify as Safe or Infected with Trojan

Figure 4.6: Process flow diagram depicting the experimental setup, methodology, and validation of the methods in this chapter.

4.4.1 Circuit Designs

We test our HT detection method using circuits implemented FPGAs to allow us to test different circuit and Trojan architectures without reasonably being able to source Trojan-infected circuits or fabricate a variety of hard-wired ASICs. The size of the trigger circuitry was shown in [25] to be one of the major factors that affected success of detecting Trojans. Using FPGAs also allows us to test multiple trigger sizes for each Trojan as seen for the AES-T1800 circuit in Figure 4.7. The relative size of the Trojan's trigger circuitry is calculated using the number of adaptive logic modules (ALMs) making up the circuit module as shown in Equation 4.3, which are the basic blocks used by these FPGAs with their usage statistics output by the bitstream compilation report [89]. We use the terasIC DE0-CV hardware design platform [90] which employs an Altera Cyclone V FPGA IC (specifically, the 5CEBA4F23C7N) [91] at its core. For the purposes of our research, we assume we have at least one uninfected device so that we can fairly compare our HT detection results with those of prior work.

$$\text{HT Size (\%)} = 100 \frac{\text{\# ALMs of trigger}}{\text{\# ALMs of uninfected circuit}} \tag{4.3}$$

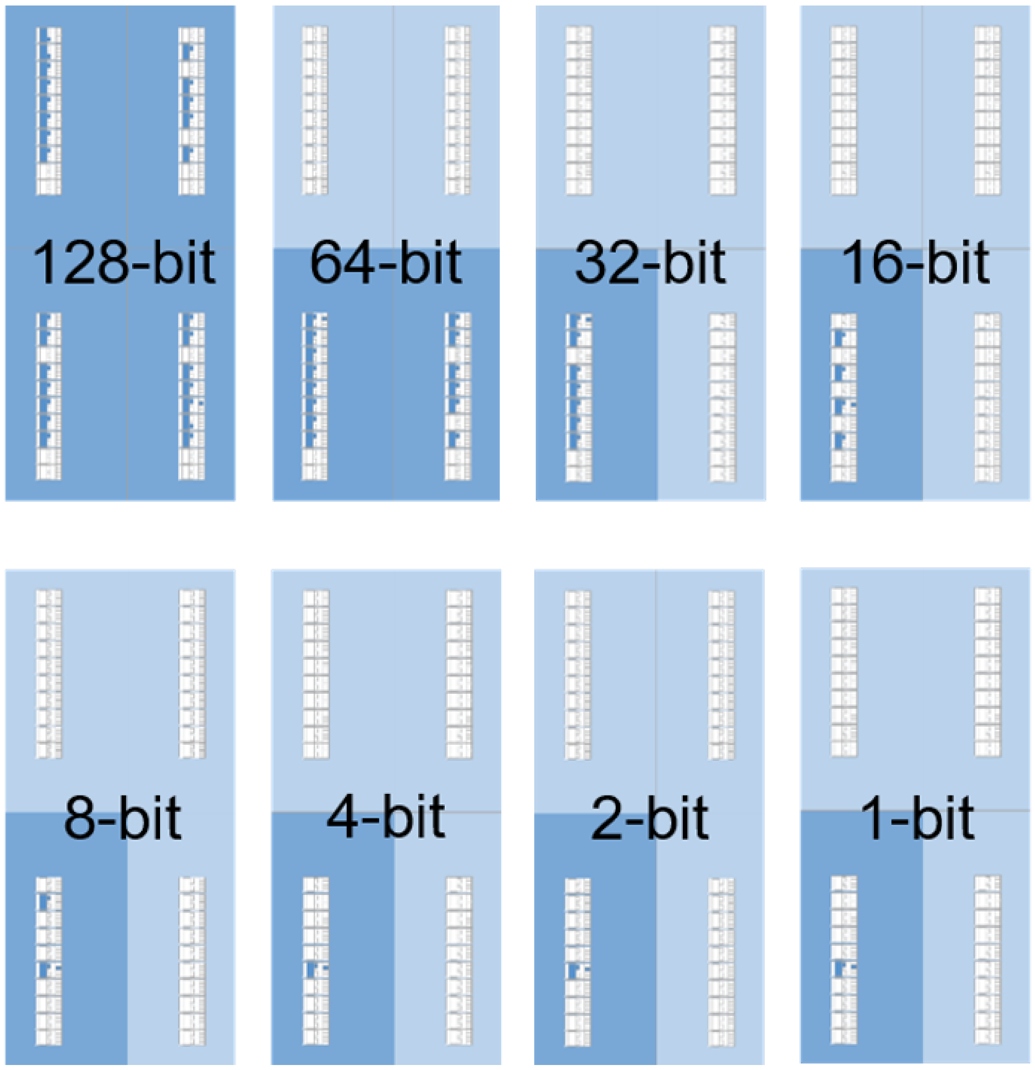


Figure 4.7: FPGA allocation of adaptive logic modules for different size triggers. The 4-, 2-, and 1-bit triggers have minute differences that are not visible at this scale.

Table 4.1: Relative size of trigger circuitry versus the uninfected circuit, measured by the number of adaptive logic modules.

Trigger Size (bits)	AES-T1800 (%)	AES-T1600 (%)	RS232-T500 (%)
128	0.701	1.298	n/a
64	0.364	0.805	n/a
32	0.182	0.493	0.600
16	0.104	0.415	0.300
8	0.052	0.260	0.150
4	0.026	0.208	0.100
2	0.026	0.130	0.050
1	0.026	0.026	0.025
Payload	0.571	2.129	1.650

We test three different circuit designs implemented on these FPGAs from the set of HT benchmarks available from TrustHub [92], [45]. Details of each circuit and Trojan are described below and relevant statistics about each are detailed in Table 4.1:

- AES-T1800: The uninfected circuit implements a cryptographic processor design that performs the 10 stages of the Advanced Encryption Standard (AES) on a 128-bit block. This circuit is infected with the T1800 Trojan whose activated payload is designed to continuously shift a cyclic shift register to consequently increase power consumption and quickly drain a battery that the device may be using. The T1800 payload is triggered by a combinatorial logic circuit that activates upon finding a specific 128-bit input to the AES circuit. We also test the same circuit with smaller trigger circuitry that monitors only 64, 32, 16, 8, 4, 2, and 1 of the least significant bits of the AES input. This circuit's combined payload and trigger circuitry make up between about 0.6% to 1.3% of the main circuit area depending on the trigger size, making them the smallest (and therefore most difficult) HTs to detect that we test.

- AES-T1600: The circuit implements the same AES circuit as above, but is injected with the T1600 Trojan. The T1600 payload is designed to leak the processor's encryption key by modulating activity on an unused output pin of the IC. The modulated activity takes advantage of the EM side channel to leak the security key through bits of information. This Trojan is activated by a trigger circuit that monitors the input for a specific sequence of three 128-bit values, rather than just one specific input as the T1800 does. We again test triggers that monitor 128-bit down to 1-bit inputs. This circuit's larger payload and complex trigger circuitry make up between about 2.2% to 3.4% of the main circuit area for different trigger sizes, making them the largest HTs that we test.

- RS232-T500: The Recommended Standard 232 (RS232) is a design standard for serial data transmission. Here, the uninfected RS232 circuit is a universal asynchronous receiver and transmitter core. The transmitter module takes 128-bit words and outputs them serially according to the RS232 standard, with the receiver doing the opposite. The T500 Trojan payload causes the transmitter to never flag when its transmission is completed, resulting in a failed transmission. The trigger monitors a 32-bit counter and activates the Trojan when the counter reaches a specific 32-bit value. We test triggers that monitor between only 32 down to 1 bit of the counter unlike the previous two Trojan circuits. The RS232-T500 circuit's payload and trigger circuitry accounts for between 1.7% to 2.3% of the main circuit area for different trigger sizes.

For each HT-infected circuit design, we compiled Verilog design source code using Altera's Quartus Prime software suite which automatically places and routes all circuit modules and connections to optimize their layout. Each circuit design is composed of a few source code modules which, when compiled into an optimized circuit layout in Quartus, allow us to view and edit specific logic cells or registers for each of the different circuit modules. For example, the infected AES-T1800 circuit is composed of four modules including an input feed, the main AES encryption circuit, Trojan trigger, and Trojan payload that can be highlighted separately within the optimized layout in the Quartus Chip Planner tool as shown in Figure 2.2. From this infected circuit layout, we use the Engineering

Change Order tool to remove only the connections and circuitry related to the Trojan and its trigger to generate the uninfected circuit which performs its task as intended. If we were to compile an uninfected and infected design separately, the software's automated placement and routing optimization would create unique designs that are potentially easily distinguished. Instead, the process of removing the specific Trojan components from the infected circuit ensures that the only differences between the uninfected and infected circuit designs are the presence of the Trojan circuitry and both circuits operate identically when the Trojan is dormant. This process allows us to emulate the addition of a covert Trojan designed to evade detection.

4.4.2 Backscattering Measurement Setup

To measure the backscattering EM side channel of the IC, we use a near-field high resolution probe [58]. The probe is fixed with a stationary 3D positioning clamp at a position close to the IC without touching it (approximately 0.5 mm about the IC, as tested in [58]) and is not moved between any measurement. The probe emits a 15 dBm sinusoid produced by a Keysight E8257D PSG Analog Signal Generator at $f_t = 3.031$ GHz to avoid interference sources in the nearby spectrum. The emitted signal reflects off the chip and is picked up by the probe's coil with 1 mm spot size, amplified by a Pasternack PE15A1010 40 dBm Low Noise Amplifier, and measured with a Keysight PXA N9030B Signal Analyzer at $H = 35$ harmonics that are multiples of the circuit's $f_c = 20$ MHz clock frequency, i.e. $3.051, 3.071, 3.091, \ldots, 3.731$ GHz. Since minor variability in the IC's clock frequency causes linearly increasing differences between the harmonic's actual and expected frequencies, we record the maximum power within 10 KHz windows surrounding each of the expected harmonic locations. This windowed capture ensure that we find the correct harmonic peak even if clock frequency jitter has moved it from its expected frequency. To capture measurements across the chip face area, we secured the DUT with four standoff screws onto a pair of Zaber X-LSQ150B stages for X-Y movement control

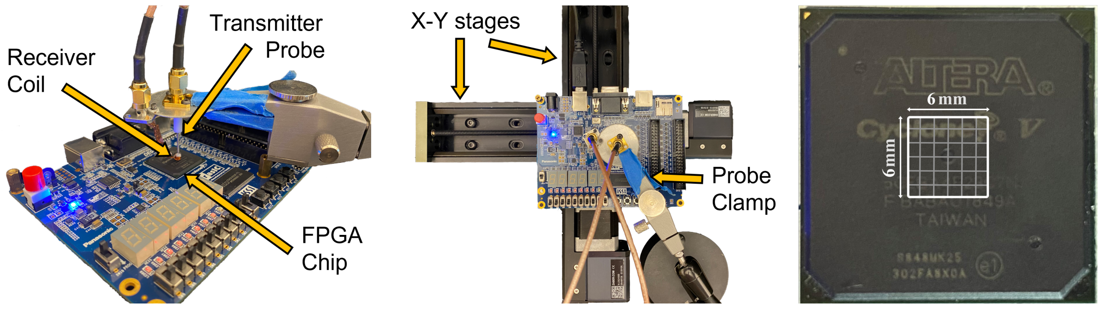


Figure 4.8: Hyperspectral scanning measurement setup with DE0-CV board and high-resolution probe (left), X-Y movement stages (middle), and FPGA IC die dimensions (right).

[93]. These move the center 6 mm $\times$ 6 mm of the chip area under the probe in 1 mm increments ($X, Y = 7$), with 50 μm accuracy and 3 μm repeatability. The measurement setup and scanning grid scale are shown in Figure 4.8. We aim to produce as repeatable measurements as possible by using high-precision movement control, affixing each DUT to the movement stages, and fixing the measurement probe at a stationary position throughout all tests.

We capture our measurements with a specific routine to ensure each measurement is captured at the same time in the circuit's operation and reduce the effect of changing chip temperature across long measurement runs. At each x, y scanning location, we first program the FPGA to run a control (uninfected) circuit and measure the signal strength at each of the H harmonics in succession. Then the FPGA is reset and programmed to run the same circuit with additional Trojan and trigger circuitry. We automatically move the IC to align the probe with the next x, y position; raster scanning across the chip in rows. This procedure results in measurements of size $\mathbf{Z} \in \mathcal{R}^{S\times X\times Y\times H}$ for each circuit programmed onto that FPGA before moving to the next copy of the FPGA board. In our tests, we capture the maximum sampling budget $S = 10$ samples at each hyperspectral point for each control circuit and S_{xyh} for each test circuit. Each hyperspectral point takes on average 1.6 seconds to capture, resulting in 7 hours or more to scan a single circuit S times for the entire measurement space.

4.4.3 Preprocessing of Measured Data

Given the measured backscattering EM side-channel data, we follow a simple two-step preprocessing procedure before analyzing our test circuit data. The first step is to convert data captured at harmonics into harmonic ratios between frequency-neighboring harmonics. This helps to mitigate the SNR differences of our measurements that may arise when vertically positioning the probe as close to the FPGA as possible. Since the data are captured as received power measurements in decibels per milliwatt, this operation is simply the difference of neighboring harmonics $\mathbf{Z}_{\text{preprocessed}} = \mathbf{Z}(:, :, :, 2{:}H) - \mathbf{Z}(:, :, :, 1{:}H-1)$. The second step is to standardize each feature to the mean and standard deviation of the control samples to allow the filtering and clustering methods we test to treat every feature equally. The resulting features are centered to zero-mean and unit-variance as seen for the frequency measurements of one x, y location in Figure 4.9.

4.5 Validation

Since prior work has shown that detecting dormant Trojans with gets more difficult as the trigger circuitry shrinks [25], we focus our experimental evaluation on scenarios with reduced trigger sizes to test the limits of our hyperspectral scanning and feature selection approaches. We first evaluate our method when trained and tested on the same board to ensure the validity of our approach. We then cross validate our method using randomly sampled measurements from ten separate DE0-CV training boards to demonstrate its robustness to manufacturing variation between copies of the same FPGA development board. Finally, we evaluate the performance of our method with tighter measurement budgets to understand the trade-offs between scanning complexity and HT detection performance.

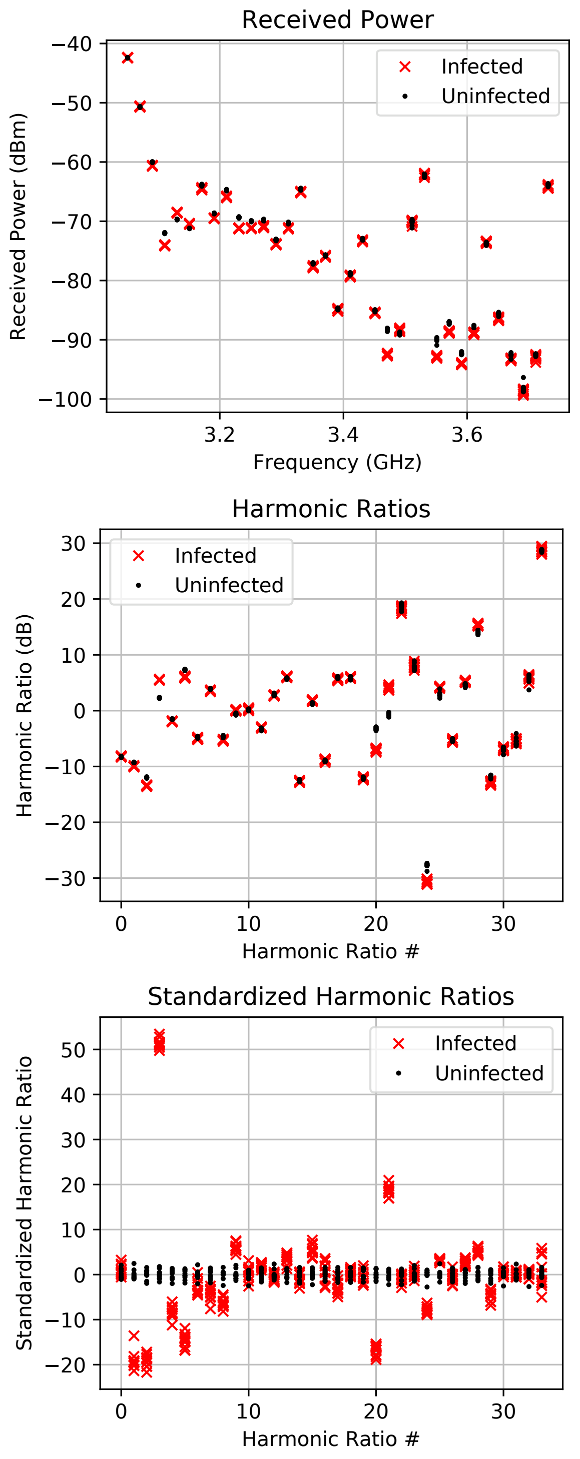


Figure 4.9: Effect of preprocessing steps on ten scans of real measurement data from the AES-T1800 circuit with 128-bit trigger, starting from the measured power for an uninfected and infected circuit across several frequencies at a single physical location on the chip (top), to power differences between neighboring frequencies (middle), to standardized power differences (bottom).

4.5.1 Single Board Performance

As a proof of concept, we first evaluate our method without taking into account manufacturing variability between copies of the same IC. We train our method using measurements of the uninfected circuits on a single board and test with separate measurements from the same board. We test each circuit 20 times while randomly swapping the order of the test scans to ensure that the results would not vary significantly over time due to transient effects such as the FPGA chip increasing temperature or other transient interference. The active learning portion of our method uses the F-test with $p_{\text{low}} = 1 - p_{\text{high}} = 0.1$ as confidence thresholds of when to decide that the test distribution matches or diverges from the control distribution to determine when to stop sampling.

Results are presented in Table 4.2 as the area under the receiver operating characteristic curve (AUC $\in [0, 1]$) to concisely demonstrate the ability of our method to distinguish uninfected from infected circuits with minimal false positives. Our method virtually perfectly detects HTs with a trigger size down to two bits for the AES-T1800 circuit, which is about four times smaller than the best baseline from Subsection 4.2.4 was able to detect. Additionally, our method detected triggers down to one bit for both the AES-T1600 and RS232-T500 circuits which correspond to about 0.03% of each circuit's area. The RS232-T500 circuit was only tested with 32-bit triggers and smaller because the trigger monitors a 32-bit counter circuit, unlike the two other circuits which monitor 128-bit states. These results clearly demonstrate that the combination of hyperspectral scanning, pre-filtering, and active learning methods is highly effective for detecting covert HTs through the backscattering EM side channel on a single DE0-CV board. To evaluate the robustness of this performance, we must also confirm that our method can detect HTs in the difficult setting that takes into account manufacturing variability across DE0-CV boards.

Table 4.2: HT detection performance as the area under the receiver operating characteristic curve (AUC) on a single DE0-CV board for each circuit with the HT trigger monitoring a given number of bits. The HT RS232-T500 monitors a 32-bit counter, so it is not tested for larger triggers. HTs are detected nearly perfectly for all Trojan types and trigger sizes, except for the smallest AES-T1800 1-bit Trojan.

Trigger Size (bits)	AES-T1800 (AUC)	AES-T1600 (AUC)	RS232-T500 (AUC)
128	1.000	1.000	n/a
64	1.000	0.999	n/a
32	1.000	0.996	1.000
16	1.000	0.992	1.000
8	0.988	1.000	1.000
4	0.991	1.000	0.997
2	0.997	1.000	1.000
1	0.845	1.000	0.995

4.5.2 Multi-Board Performance

While the differences between an uninfected and infected circuit are detectable at almost any tested trigger size for a single board, the manufacturing variations between copies of the DE0-CV boards are larger as seen in Figure 4.5 and may obscure smaller Trojans. Here we train across nine copies of the DE0-CV board loaded with the uninfected circuit before testing a corresponding circuit on a tenth board. This tenth board is then loaded with either the uninfected or infected circuit and is tested against the distribution of training boards using the distance metric in Equation 4.2. Each circuit was again tested 20 times with random swapping of the scan order and $p_{\text{low}} = 1 - p_{\text{high}} = 0.1$ for the measurement confidence stopping criteria in the active learning method.

While slightly worse than when testing with a single DE0-CV board as expected, the results in Table 4.3 demonstrate that our hyperspectral scanning and feature selection method can detect HTs even when taking into account some manufacturing variability between boards. Our method virtually perfectly detects HTs with a trigger size down to four bits for the AES-T1800 circuit, which is two times smaller than the best baseline from Subsection 4.2.4 as well as about 14 times smaller than prior work was able to detect when

Table 4.3: HT detection performance (AUC) trained on multiple DE0-CV boards for each circuit with the HT trigger monitoring a given number of bits. HTs are detected more consistently than prior work [25] for the majority of tested circuits, but expectedly less well than when trained on a single board.

Trigger Size (bits)	AES-T1800 (AUC)	AES-T1600 (AUC)	RS232-T500 (AUC)
128	1.000	0.998	n/a
64	1.000	0.995	n/a
32	1.000	1.000	1.000
16	1.000	0.989	0.973
8	0.983	0.998	0.993
4	1.000	1.000	0.940
2	0.873	1.000	1.000
1	0.498	1.000	0.891

scanning at a single location. While prior work did not test versions of the AES-T1600 and RS232-T500 circuits with smaller triggers, we show here that we can detect triggers down to 2 bits in the AES-T1600 circuit and down to 1 bit in the RS232-T500 circuit. Although this peak performance greatly improves on the results of previous work, our method of hyperspectral scanning requires more scanning time.

4.5.3 Sampling Reduction

Finally, we test the performance of our method across a range in the number of point-scanned measurements for a sample to analyze the trade-offs between reduced sampling and HT detection for the different circuits and trigger sizes. It is important to note that three factors affect the total number of measurements captured for a specific circuit. The first is the number of features that are pre-filtered out by their ranking from highest to lowest standard deviation in the training set, according to the method in Subsection 4.3.1. By removing a specific number of features from the set to be scanned, we have a deterministic way to reduce the feature set's size.

The second factor that affects the number of measurements taken is the confidence threshold used for the active learning portion of our method. Varying p_{low} and p_{high} has a

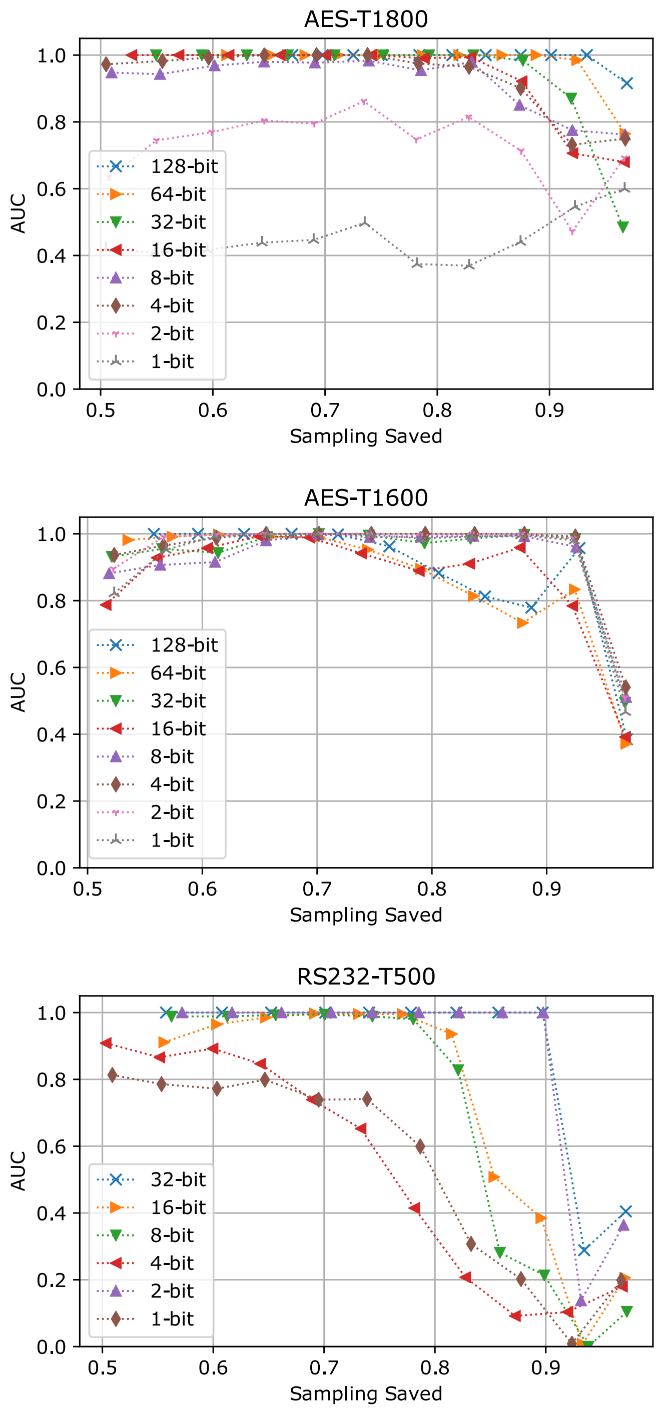


Figure 4.10: HT detection performance for three tested circuits as the number of features filtered out from the sampling set increases (resulting in less sampling from left to right).

relative effect on the reduction of the feature set size. For instance, a very small p_{low} or very large p_{high} imply the need for greater confidence in the measurement difference from, or similarity to, the control measurement's distribution, respectively. To achieve greater confidence, more measurements must be taken at a single hyperspectral point and consequently requires more scanning across the whole hyperspectral space to be statistically confident at all locations. Conversely, choosing loose bounds on these thresholds (where p_{low} and p_{high} approach 0.5 from below and above, respectively) allows our method to scan a single time at each hyperspectral point before moving to another. Unlike the first factor, this is a relative factor and will affect the number of scans depending on the actual circuits, measurements, etc. Specifically, we find in our tests that HTs with larger triggers are classified with the fewest samples because they differ from the control measurements most confidently.

The third factor is the scanning budget S which can directly affect the number of scans performed, particularly when p_{low} and p_{high} are chosen to require great confidence in the measurement's difference or similarity with the control. With tight confidence thresholds, it is possible that all features in the hyperspectral space will be scanned indefinitely. Therefore S bounds the maximum number of total scans across the hyperspectral space to $S{\cdot}X{\cdot}Y{\cdot}H$, or simply S at each location.

Understanding those three factors, we test the performance of our method across a range of total samples captured. For simplicity, we again analyze the case where $p_{\text{low}} = 1 - p_{\text{high}} = 0.1$. We choose $S = 10$ because we find it is large enough to not restrict sampling before most measurements can be confidently classified. With these two factors set, we vary the number of features filtered out from the sampling set to analyze the performance versus sampling savings trade-off. Here we scale the sampling savings to the range $[0, 1)$, where zero represents sampling the entire hyperspectral space S separate times and one represents no sampling at all. Results for each circuit and trigger size are presented in Figure 4.10.

As can be seen for almost every circuit and trigger size, an optimal level of sampling occurs when around 60 to 75% of potential sampling is saved. We find some agreement with our hypothesis that scanning the entire hyperspectral space can add unnecessary noise that distracts from detecting the minute signal differences caused by the presence of a Trojan. This is most apparent for the AES circuits which improve detection as the sampling savings increases. However, detection performance begins to suffer in most cases without sufficient sampling, as can be seen by the dip in AUC when approaching a sampling savings of one. This phenomenon tells us that our method of removing features as ranked by their standard deviation across training samples as in Subsection 4.3.1 matches well with the ideal performance when removing features known not to distinguish Trojans as in Figure 4.10 for modest sampling reduction.

4.6 Conclusions

This work, published in [29], established a novel non-destructive method of detecting hardware Trojans in ICs up to 14 times smaller than prior work by selectively measuring the backscattered EM side-channel signal across space and frequency. We develop four baselines of standard or idealized strategies for Trojan detection in this setup against which we compare our validated results favorably. Our feature selection method significantly outperforms the results obtained from simply measuring the entire hyperspectral space or a noise-reduced version of it, and even surpasses the best possible results achieved when sampling all frequencies at a single chip location. This paper demonstrates the need to measure a selected subset of features distributed across the hyperspectral space to detect Trojans as small as 0.03% of the circuit size that are normally obscured by noise, interference, and manufacturing variabilities between ICs.

The strong performance of the methods developed in this work motivates further investigation into feature selection methods and hyperspectral scanning of the backscattering EM side channel for hardware Trojan detection. Future work is needed to demonstrate the success of these methods for other Trojan and trigger varieties that do not scale in size like the internally-activated, condition-based HTs we tested here. Additional study would be useful to analyze the robustness to manufacturing variabilities for other ICs, especially for ASICs or those with higher transistor density. Further maturation of these methods to reduce hyperspectral sampling time allowing higher testing throughput is an open area of research. While this work assumes the possession of at least one uninfected device, an extension of this work may demonstrate these methods without that assumption by using circuit simulations, one class classification, anomaly detection, or other methods.

CHAPTER 5
HYPERSPECTRAL IMAGE RECOVERY VIA RELIABILITY-WEIGHTED COMPRESSED SENSING FOR HARDWARE TROJAN DETECTION

5.1 Overview

With the recent attention focused on detection of dormant hardware Trojans [94], [25], [29], it is important to keep in mind the real-world application of these strategies. Destructive techniques, although thorough and robust for detecting hidden Trojans, require large time and human capital investments to validate a small proportion of manufactured devices or the integrated circuits on them. Non-destructive methods until recently were only viable for detecting active Trojans but could be performed at higher throughput since they could be more easily automated. The ability to non-destructively detect dormant Trojans remained elusive until the work of [25] whose impedance-based electromagnetic side channel method could detect large dormant Trojans. Significant improvements were made in [29] using hyperspectral measurements of the same side-channel signals to detect much smaller Trojans than had previously been achieved. In doing so, that work traded a much more significant measurement time to lead to state-of-the-art detection performance in terms of Trojan size detected non-destructively. While this method no longer requires destroying the device under test, its measurement times put it on a similar scale of testing throughput as destructive methods.

To improve upon the performance of backscattering EM side-channel techniques for non-destructive Trojan detection while heavily reducing the measurement requirements for hyperspectral scans necessary to do so, we develop a compressed sensing strategy to quickly recover hyperspectral images of an IC for efficient and robust HT detection. Our novel strategy weights random sampling of the hyperspectral space toward features that are known to be less consistent across a training set of ICs. Focusing on these unreliable features allows our reconstruction to quickly account for the peculiarities of a specific IC with few measurements to reconstruct the full hyperspectral images more accurately and consequently improve HT detection. We evaluate this novel sampling method as compared to prior work on a benchmark circuit infected with a Trojan. We test our method against traditional uniform sampling across three different reconstruction bases and show that we can uncover dormant hardware Trojans with state-of-the-art accuracy and as little as one tenth of the measurements of previous work.

5.2 Dormant Hardware Trojan Detection

5.2.1 Compressed Sensing

A popular way of reducing measurement costs is to reconstruct images from very few measurements in a compressed sensing framework. Unlike iteratively selecting individual measurements as in an active sampling or reinforcement learning regime, CS takes advantage of an assumed underlying structure in the data's domain to allow sparse random sampling for reconstruction. Many works used basis functions like the Fourier transform or discrete cosine transform (DCT) to approximate the smoothness and continuity of the data's domain. For better reconstruction of natural images, other bases were developed that incorporated piece-wise smooth properties like various wavelet functions [60]. Further work showed that an overcomplete dictionary of bases learned from images similar to those being reconstructed often outperformed any of the former bases for which closed form analytical expressions exist [61], [62].

As introduced in Section 2.5, the CS framework can be represented with the standard linear model $\mathbf{y} \approx A\mathbf{x}$ where $\mathbf{y} \in \mathbb{R}^M$ is a vector of measurements, $\mathbf{A} \in \mathbb{R}^{M \times N}$ is a measurement sampling matrix and $\mathbf{x} \in \mathbb{R}^N$ is a set of learned coefficients that represent the reconstructed image. Assuming $\mathbf{x}$ can be sparsely represented by some set of D basis vectors $\mathbf{\Psi} \in \mathbb{R}^{N \times D}$, then this problem can be rewritten as $\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{\Psi}\mathbf{s}$ where $\mathbf{s} \in \mathbb{R}^D$

is the sparse coefficient vector reconstructing $\mathbf{x}$ in that basis, meaning $\mathbf{x} = \mathbf{\Psi}\mathbf{s}$. To ensure that the solution can be stably recovered, $\mathbf{A}$ and $\mathbf{\Psi}$ should be incoherent, meaning they cannot sparsely represent the other. In practice, $\mathbf{\Psi}$ could be the Fourier, Wavelet, learned dictionary, or any other bases that are sparsifying for the signals to be reconstructed. $\mathbf{A}$ is then incoherent with $\mathbf{\Psi}$ if it is structured for uniformly-random sampling. This binary matrix $\mathbf{A}$ is used in this context to measure a sparse random set of pixels which are then used to reconstruct the entire hyperspectral image image. In the standard setting where $M > N$, this problem can be solved with least squares and has a unique minimum. However, in the heavily underdetermined setting where the number of samples is far fewer than the dimensionality of the data ($M \ll N$), the sparse recovery problem shown in Equation 2.1 is required to be used. Although not as simply proven, recent work has shown that the solution to this problem is still unique and can be solved with a standard LASSO solver [63], [64].

Hyperspectral imaging has received significant attention as an application of compressed sensing due to the measurement time required to acquire hundreds of images across a range of frequencies. As noted in [29], point-scanning to capture hyperspectral images further lengthens the full image capture time. Point-scanning traditionally entails raster-scanning a measuring device spatially in two dimensions while measuring the response at specific frequencies, one at a time, and can potentially take hours per scan depending on the resolution and dimensions of the desired scan. Since antenna arrays that can capture high resolution spatial information of the electromagnetic side channel all at once do not exist, this point-scanning approach is unavoidable. To reduce scanning time in the point-scanning HSI regime, we turn to the compressed sensing. Here we utilize the CS framework to efficiently recover hyperspectral images through the backscattering EM side channel without having to measure every single hyperspectral location.

5.2.2 Backscattering EM Side Channel

The standard electromagnetic side channel arises as a leakage of information caused by variations in the current running through conductive traces on or within devices and the integrated circuits placed on them [10], [47]. As stated previously, EM side-channel signals show up as spikes in the frequency domain due to the device's clock signal or other repetitive activity. Like the power side channel that is sometimes used to monitor integrated circuits, the EM side channel is current-based so it varies depending on program and circuity activity on the IC [9]. To find hidden malicious circuitry that may not draw enough electrical current to be detected with other side-channel methods, the authors of [25] introduced the backscattering EM side channel. The backscattering EM side channel is created by transmitting a continuous wave sinusoid toward an IC and measuring the backscattered signal. This reflected signal is modulated depending on the state of underlying transistors and yields information about the IC's architecture based on the impedance state of those transistors [50].

When a signal with frequency f_t is transmitted toward the IC, transistors switching at the device's clock frequency f_c modulate the incoming signal and the resulting backscattered signal can be seen as spikes and harmonics in the frequency domain at $f_t \pm f_c, f_t \pm 2f_c, \ldots$ as shown in Figure 2.3. Since EM side-channel signals will naturally vary across the face of an IC due to its architecture, it is crucial to probe the spatial area of the IC to find hidden malicious hardware modifications. Demonstrating the importance of measuring signals across the IC, the authors of [29] used full, point-scanned hyperspectral measurements to uncover dormant HTs up to 14 times smaller at the cost of a significant increases in measurement time. In this work, we use the flexibility of the hyperspectral point-scanning approach to capture a much sparser set of measurements than previous work which can then be used within the CS framework to reconstruct full hyperspectral images of the backscattered EM side-channel signal. With the reconstructed hyperspectral images we detect the same hidden malicious circuitry injected into an IC known as hardware Trojans.

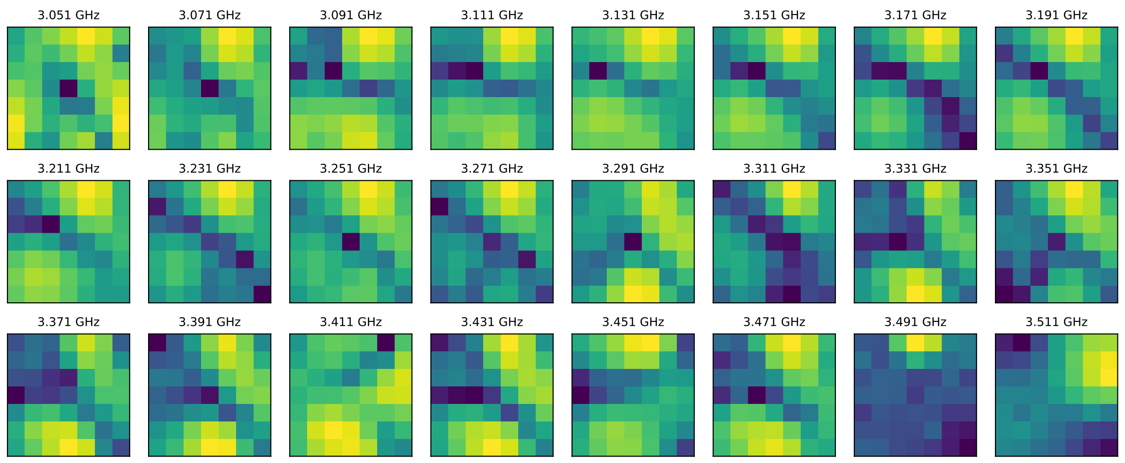


Figure 5.1: Normalized spatial measurements of the backscattered EM side-channel signal for the first 24 harmonics above the incident 3.031 GHz signal. Harmonics generally exhibit smooth spatial variation. Variation between between frequencies is generally less smooth at higher harmonics.

5.3 HT Detection with Compressive Hyperspectral Scanning

Chapter 4 demonstrated a novel method to capture hyperspectral images of backscattering EM side-channel emanations by raster-scanning a probe across the face of an IC with automatically controlled movement stages to improve hardware Trojan detection accuracy greatly [29]. While we do not necessarily measure emanations at every spatio-spectral location in this work, we follow a similar setup by measuring the peak received power at hyperspectral points defined by their physical location on the chip (x, y) and harmonic index (h) above the backscattered center frequency. However, rather than raster-scanning at all chip locations, we instead save significant measurement time and improve robustness by selectively point-scanning the hyperspectral space with our novel reliability-weighted CS methodology as detailed in this section.

5.3.1 Hyperspectral Image Recovery with Reliability-Weighted Sampling

Prior work has shown relatively smooth variation of the backscattered EM side-channel signal spatially across the face of an IC [29], but that is not necessarily the case across the frequency dimension as shown in Figure 5.1. While many harmonics exhibit smooth transitions across space and frequency, several harmonics seem to sporadically differ significantly from their spectrally-neighboring counterparts, such as at 3.051 GHz, 3.291 GHz, and 3.491 GHz in the figure. For this reason, we break the reconstruction problem up into H separate 2D image reconstruction problems $\mathbf{y}_h \approx \mathbf{A}_h \mathbf{x}_h$ where H is the number of harmonics that are measured. When capturing a total of M samples across the entire hyperspectral space, we sample M_h points at each frequency and only reconstruct an image when $M_h \geq 2$ so the problem is well posed. Sampling M_h points results in the measurements $\mathbf{y}_h$ and measurement matrix $\mathbf{A}_h \in \mathbb{R}^{M \times X \cdot Y}$ where X, Y are the number of horizontal and vertical positions we can scan on the IC, even though not all of them may be scanned. The measurement matrix $\mathbf{A}_h$ is formed so each column is zero everywhere except with a one at the position corresponding to the vectorized index x, y of that sample. For sufficiently large M, we may sample the same point more than once in which case we update our existing measurement in $\mathbf{y}_h$ to the mean of the repeated measurements at that point.

To choose the locations on the IC and their corresponding frequencies that are measured, we sample from the three-dimensional discrete distribution $p(x, y, h)$. For standard uniformly-random sampling of the hyperspectral space, each consecutive sample is chosen from the uniform distribution $p_{\text{uniform}}(x, y, h) = \frac{1}{X \cdot Y \cdot H}$. Here we sample with replacement because it can be useful to have multiple samples of the same point when those points have high noise or large variance between ICs. On average, sampling uniformly will yield the same number of samples per harmonic so each reconstruction problem will be relatively similar in terms of complexity.

Prior work has shown that the backscattering EM side channel may vary significantly between copies of the same IC [25], [29]. While some regions in the hyperspectral space may look similar across all ICs, the regions that look different can hold information that may reveal whether the IC is infected with an HT or simply varies due to the natural imperfections of the manufacturing process. The hyperspectral locations that reliably show statistically significant differences between an uninfected and infected IC may be very sparse for small Trojans. This can be as few as 0.2% of locations for small, dormant Trojans as shown in Chapter 4. If we simply capture samples uniformly at random as is the standard approach to compressed sensing problems of this type, these locations are likely to go undiscovered. To maintain the advantages of a randomized sampling routine required for CS reconstruction while also discerning distinguishing features of the hyperspectral space, we develop a reliability-weighted sampling strategy.

Instead of capturing samples with even likelihood across space and frequency, our reliability-weighted sampling strategy focuses measurement effort where we expect to quickly discern the distinct features of an individual IC. We do so by weighting our random sampling distribution toward the hyperspectral locations that vary significantly across ICs. This follows the hypothesis that locations in the hyperspectral space that consistently reflect EM signals similarly across a variety of ICs (reliable features) will be unlikely to distinguish the minute variations caused by the presence of a Trojan. Conversely, we expect that the locations that commonly differ between ICs (unreliable features) are more likely to display distinguishing features when an IC is infected with a Trojan and may require multiple measurements to distinguish confidently. For those reasons, we use the reliability of measurements as a metric with which we can weight sampling the sampling distribution more heavily toward locations that are expected to be unreliable and potentially tell us the most information about the presence of a Trojan. Of course, with the wide variety of circuits and Trojans that exist and their unknown backscattering effects, we do not completely halt measurements of reliable locations. All locations in the hyperspectral space have a sampling probability greater than zero in our weighted sampling strategy to account for this.

To measure the reliability of individual features in the hyperspectral space, we compare the spread of samples across different known-uninfected ICs against the spread of samples for individual uninfected ICs. Given hyperspectral measurements $\mathbf{Y}_j \quad \forall j \in \{1, 2, \ldots J\}$

of each uninfected IC j, we compute a reliability ratio $\mathbf{E} \in \mathbb{R}^{X \times Y \times H}$, where each point is

$$\mathbf{E}_{xyh} = \frac{Var(\{\mathbf{Y}_{1,xyh}, \mathbf{Y}_{2,xyh}, \dots \mathbf{Y}_{J,xyh}\})}{\frac{1}{J}\sum_{j=1}^{J} Var(\mathbf{Y}_{j,xyh})}, \tag{5.1}$$

and large values indicate that samples of a specific location x, y, h are highly variant across ICs (unreliable) but minimally variant for individual ICs. Small values indicate the most reliable locations which would have an approximately equal variance across the measurements of different ICs as they do for the measurement variance for the individual ICs. We convert that ratio to our reliability-weighted sampling distribution by scaling the values to sum to one, i.e.

$$p_{\text{reliability}}(x, y, h) = \frac{\mathbf{E}_{xyh}}{\sum_{xyh} \mathbf{E}_{xyh}}. \tag{5.2}$$

The discrete sampling distribution generated by computing the reliability of our hyperspectral backscattered EM side-channel measurements is pictured in Figure 5.2. Comparing with the measurements themselves in Figure 5.1, we can see that features tend to be the least reliable (and thus sampled with greatest likelihood) at local minima for each scanned frequency. This is exemplified especially by the first five harmonics (3.051 - 3.131 GHz) in Figure 5.2 which have a sparse set of the least reliable features across the entire hyperspectral space. In contrast, a few frequencies such as 3.151, 3.191, and 3.391 GHz exhibit relatively reliable features across the spatial dimension, while others such as 3.311 and 3.411 GHz are generally unreliable overall.

5.3.2 CS Recovery Bases

Traditional compressed sensing as it was designed used orthonormal bases $\mathbf{\Psi}$ like the Fourier or Wavelet bases that have closed form analytical expressions to develop a hierar-

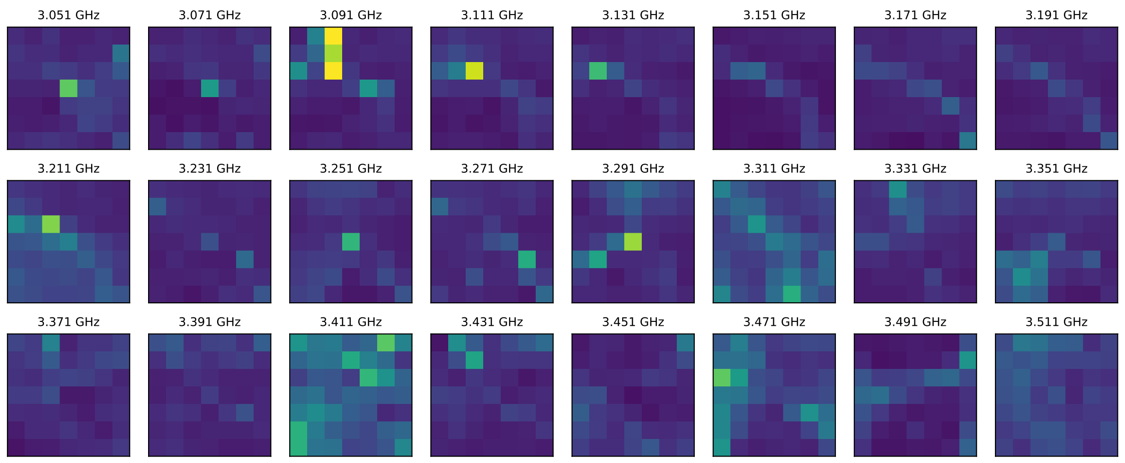


Figure 5.2: Reliability-weighted sampling distribution $p_{\text{reliability}}(x, y, h)$ of the backscattered EM side-channel signals for the first 24 frequencies above the incident 3.031 GHz signal. Larger values correspond to hyperspectral locations that will be sampled with greater likelihood due to the features' lower reliability across ICs there. Sampling emphasis is generally focused toward local minima of the backscattered EM signal for each frequency.

chy of basis functions [59]. These bases allowed nice theoretical properties that in certain situations could yield probabilistic guarantees on recovery error bounds. More recent work showed the value of using learned overcomplete dictionaries for image recovery [61], [62]. While these dictionaries lost the hierarchical structure of the aforementioned orthonormal bases, their recovery performance warranted the tradeoff. Here we test image recovery with both the two-dimensional orthonormal discrete cosine transform basis functions and also dictionaries of 2D bases learned from the measured signals of uninfected ICs.

The two-dimensional DCT basis we use here is specifically the DCT-II basis which is commonly used for JPEG image compression [95]. The definition for the first $D = K^2$ two-dimensional DCT basis functions is given in Equation 5.3) where $\otimes$ is the Kronecker product of two vectors and the leading coefficients make the DCT an orthogonal set of bases. Here we use $K = 7$ to generate 49 DCT bases that correspond to the size of our measurement grid, unlike the traditional 8×8 bases used for JPEG compression.

$$\psi_{k_1,k_2}(x,y) = \frac{2\eta_{k_1}\eta_{k_2}}{K} cos\left(\frac{\pi k_1(\mathbf{x}+\frac{1}{2})}{K}\right) \otimes cos\left(\frac{\pi k_1(\mathbf{x}+\frac{1}{2})}{K}\right) \tag{5.3}$$

$$\forall k_1, k_2 \in \{1, 2, \ldots, K\}$$

$$\eta_k = \begin{cases} \frac{1}{\sqrt{2}}, & k = 0 \\ 1, & \text{otherwise} \end{cases}$$

Unlike the DCT bases, a learned dictionary can have an arbitrary number of D basis functions as determined by the user. The learned dictionary bases are generated from the sparse optimization problem given in Equation 5.4 where $\mathbf{U}$ is the matrix that combines the learned dictionary basis functions $\mathbf{\Psi}$ to approximate the training data $\mathbf{Y}$. The Frobenius norm "Fro" is the element-wise $L2$ norm of a matrix that forces the encoding matrix and dictionary to approximate the training data, while the element-wise $L1$ norm is used to ensure the dictionary can sparsely encode the data by approximately minimizing the number of nonzero elements in $\mathbf{U}$. The optimization problem is solved with an alternating minimization scheme, where the optimizer alternates between updating the dictionary $\mathbf{\Psi}$ with a fixed sparse code matrix $\mathbf{U}$ and fixing the sparse code while updating the dictionary. In this case, the training data is the entire set of spatial measurements of the backscattered EM signal across all frequencies for known-uninfected ICs. Examples of the DCT and learned dictionary basis functions are in Figure 5.3.

$$\min_{\mathbf{U},\mathbf{\Psi}} \|\mathbf{Y} - \mathbf{U}\mathbf{\Psi}\|^2_{\text{Fro}} + \alpha\|\mathbf{U}\|_1 \tag{5.4}$$

$$\text{subject to } \|\mathbf{\Psi}_d\|_2 \leq 1 \quad \forall d \in \{1, 2, \ldots, D\}$$

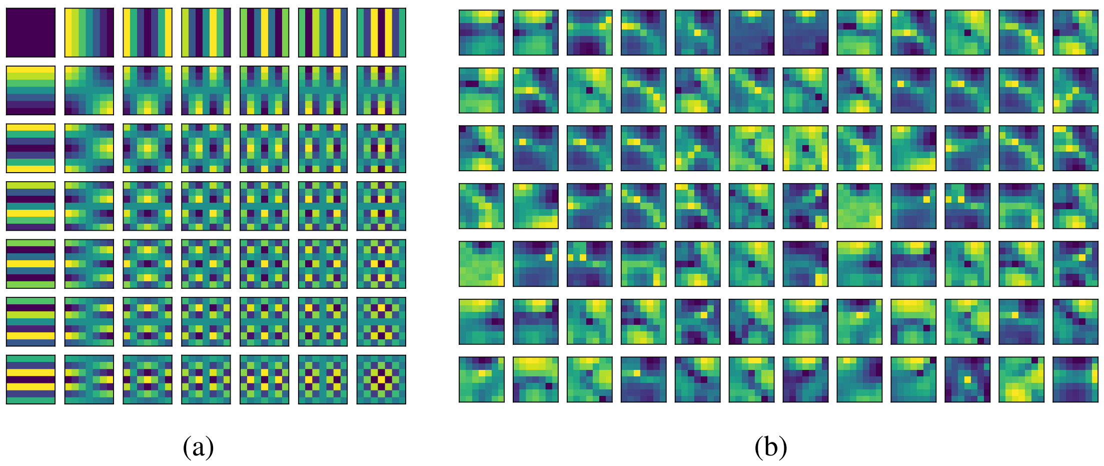


Figure 5.3: Example 7x7 two-dimensional basis functions for the (a) DCT and (b) dictionary bases learned from spatial slices of hyperspectral backscattering side-channel images captured of uninfected ICs.

5.3.3 Dormant Trojan Detection

As outlined in Chapter 4, it is possible that less than 0.2% of features in the hyperspectral backscattered EM measurement space may show a difference between an uninfected and infected circuit for certain Trojans. With such small differences, the sparse set of features that signal the presence of a Trojan may easily be hidden by the overwhelming majority of features that do not. Without knowing the location of an HT *a priori*, we follow the same feature filtering process and HT detection algorithm as [29] to decrease the dimensionality of the feature space and reduce the amount of noise present overall in the hyperspectral measurements. Since we do not know which features will be most likely to distinguish an infected IC from an uninfected one, we use an unbiased metric to measure the distance between the ICs' measurements. Given the real hyperspectral measurements of all the known-uninfected ICs in a control set $\{\mathbf{Y}_1, \mathbf{Y}_2, \ldots \mathbf{Y}_J\}$, we measure the total $L1$ distance between the average control measurements and the reconstructed hyperspectral image from measurements of a device under test (DUT). To detect an infected IC, we first generate a distribution of the distance from one control sample to the average of the rest of the control ICs. Then, if the distance of a DUT's reconstruction to the average control samples differs

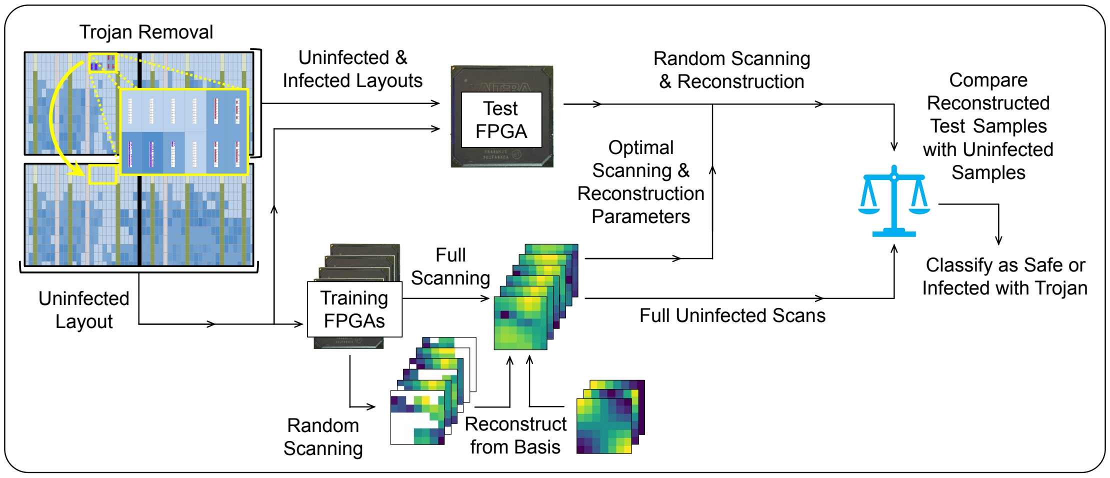

Trojan Removal
Uninfected & Infected Layouts
Test FPGA
Random Scanning & Reconstruction
Compare Reconstructed Test Samples with Uninfected Samples
Optimal Scanning & Reconstruction Parameters
Classify as Safe or Infected with Trojan
Uninfected Layout
Training FPGAs
Full Scanning
Full Uninfected Scans
Random Scanning
Reconstruct from Basis

Figure 5.4: Flow diagram depicting the circuit layout design, random hyperspectral sampling, image reconstruction, and Trojan detection process in this chapter.

by more than some threshold for a designated confidence level, it is predicted to be infected.

5.4 Trojan Design and Hyperspectral Measurement Setup

We design our experiments to test the performance of our methods for detecting dormant Trojans while also reducing measurement costs. First, we design the test circuits to have as minimal a difference between infected and uninfected circuits as possible to simulate a stealthy Trojan. Next, we develop a measurement setup for repeatable and consistent measurements of the hyperspectral space. Finally, we build a CS framework incorporating validation with multiple reconstruction bases to ensure the robustness of our analysis. The entire sampling and reconstruction process is depicted in Figure 5.4.

5.4.1 Circuit Design

We develop the uninfected and infected circuits as circuit logic modules to be loaded onto FPGAs to allow testing without requiring the fabrication of customized ICs. This also allows us to test a variety of Trojan sizes by simply loading a new bitstream to the FPGA.

To generate the circuit bitstream, we compile Verilog source code in Altera's Quartus Prime software suite that automatically routes all circuit modules and connections into a

Table 5.1: Relative size of trigger circuitry compared to the resources used by the full uninfected circuit, measured as the number of adaptive logic modules.

Trigger Size (bits)	AES-T1800 Circuit Size (%)
128	0.701
64	0.364
32	0.182
16	0.104
8	0.052
4	0.026
2	0.026
1	0.026
Payload	0.571

resource-optimize layout. The circuit design is composed of circuit modules which can be edited down to the scale of individual logic cells, connections, or registers. Given an infected circuit layout, we again use the Engineering Change Order tool to remove the individual circuit cells containing the Trojan and trigger. This allows us to generate a clean circuit which simply performs AES encryption as expected in normal operation. This Trojan removal ensures that the only differences between the Trojan-infected and uninfected circuits are the Trojan circuitry.

We test our methods on the benchmark AES-T1800 crytopographic processor design from TrustHub [92], [45]. As detailed in Chapter 4, this circuit's intended design is to perform the 10 stages of 128-bit encryption with the Advanced Encryption Standard (AES). This layout is infected with the T1800 Trojan which is designed to make continuous cyclic shifts in a register to heavily increase power consumption and quickly drain a connected battery. The overall circuit contains four modules: an input feed, encryption blocks, Trojan trigger, and Trojan payload. That Trojan is activated by a logic circuit which monitors to find a specific 128-bit input to the circuit. We test this circuit with trigger circuitry of reduced sizes to directly compare with prior work [29], [25] which found that reduced trigger sizes were what made HT detection most difficult. These reduced triggers monitor only 64, 32, 16, 8, 4, and 2 of the least significant bits of the input instead of the full 128 bits originally designed. This circuit's trigger circuitry makes up between about 0.02% to 0.7% of the AES circuit area for different trigger sizes as seen in Table 5.1. These sizes are measured using the number of basic logic blocks in the FPGA platform, known as adaptive logic modules according to the FPGA's documentation [89].

Finally, we use ten copies of the terasIC DE0-CV FPGA platform [90] with an Altera Cyclone V IC (5CEBA4F23C7N) [91]. We assume that our FPGA platforms have identical IC models that are uninfected with physical Trojans so that we may implement the Trojans ourselves through the FPGA bitstream. We expect that the ICs have some physical differences so we put forth significant effort to ensure these methods hold in the presence of significant manufacturing variabilities of the hardware.

5.4.2 Measurement Setup

We follow a very similar measurement setup as is used in Chapter 4 to be able to compare results directly. Shown in Figure 4.8, we use the same high resolution near-field probe [58] to emit and measure the backscattering EM side-channel signals. The probe is fixed with a stationary clamp about 0.5mm above the face of the FPGA IC and is not moved throughout any of the tests. The probe tip emits a 15 dBm sinusoid at $f_t = 3.031$ GHz which is produced by a Keysight E8257D PSG Analog Signal Generator. The emitted signal backscatters after contacting the chip and is received by the probe coil within an approximately 1 mm spot size. The received signal is then amplified by a Pasternack PE15A1010 40 dBm Low Noise Amplifier and measured with a Keysight PXA N9030B Signal Analyzer.

For each measurement of a random hyperspectral location to be taken, we use the desired x, y, h coordinates to scan an individual location in the hyperspectral space. To scan a specific x, y location, we move the device under the probe to the specified point with two Zaber X-LSQ150B stages [93]. The device is securely fixed to the stages with standoff screws at each corner to be repeatable within the error tolerances of the board's manufacturing and minimal stage adjustment errors. The stages handle precise X-Y movement control to center the probe onto one of 49 points ($X, Y = 7$ with 1 mm increments) on the center 6 mm $\times$ 6 mm of the chip area. To scan a specific frequency when there may be some amount of frequency jitter for the harmonics, we measure a 10 KHz frequency window surrounding one of the $H = 35$ harmonics at multiples of the FPGA device's $f_c = 20$ MHz clock frequency above the probe's emitted frequency. The maximum power within that frequency window is recorded as the measurement $\mathbf{X}_{xyh}$ for the DUT. Those measurements are then combined in the CS framework to reconstruct full hyperspectral images.

5.5 Experimental Validation

We validate our method for HT detection performance with thorough randomized testing and also analyze the effect of our weighted sampling method for compressed sensing image reconstruction. First, we demonstrate the ability of our CS approach to heavily reduce measurement requirements while matching or improving upon HT detection results of prior work. We then analyze the CS reconstruction performance of our weighted sampling strategy compared with standard uniform sampling for the DCT and learned dictionary bases.

5.5.1 HT Detection Performance

We use measurements from ten separate DE0-CV devices to account for manufacturing variabilities between ICs with our reliability-weighted sampling method. For each of these devices, we measure ten scans of the entire hyperspectral space to use as our uninfected control set. From these samples, we devise a simple reconstruction problem to find optimal hyperparameters for reconstructing a hyperspectral image at test time. For each of the J training devices, we set up the same reconstruction problem $\min_{\mathbf{s}} \|A\Psi\mathbf{s} - \mathbf{y}_j\|_2 + \alpha\|\mathbf{s}\|_1$ where $\mathbf{y}_j = \frac{1}{10}\sum_{i=1}^{10} \mathbf{Y}_{j,i}$. Instead of reconstructing individual samples directly, we reconstruct the average of all ten samples $\mathbf{Y}_{j,1:10}$ for that device across 21 log-scaled values for the range of the sparsity parameter $\alpha \in [10^{-4}, 10^{0}]$. Assuming that the average of samples is a better approximation of the underlying signal without noise, reconstructing the sample average allows us to choose model parameters that best approximate a noise-free version of the samples, rather than fitting to individual noisy samples. We choose the α with lowest average reconstruction error across 50 random trials; each reconstructing samples from the 10 devices and yielding 500 total trials. We also test with the number of random measurements ranging between 100 and 5000 in increments of 100. The upper limit of 5000 is approximately the number of measurements used to obtain the results in prior work [29]. We similarly choose the best number of measurements to reconstruct the training samples across the same 500 random trials. Finally, the reconstruction parameters are trained in this way for three unique sets of bases. These are the 49 DCT basis functions pictured in Figure 5.3a, and two sets of learned dictionary bases. The first dictionary of 50 basis functions is learned from the 3500 two-dimensional slices of the training device samples (35 harmonics $\times$ 10 devices $\times$ 10 samples). The second dictionary has 100 basis functions of the same size learned from the same training data.

Trojan detection results are presented in Table 5.2 as the area under the receiver operating characteristic curve (AUC $\in [0, 1]$). An AUC of one signifies the ability to perfectly distinguish all samples between uninfected and infected DUTs and 0.5 means the DUTs are indistinguishable. The result in the table were obtained from 50 random reconstruction trials of each trigger size and basis type, reconstructing with the parameters trained as described previously in this section. As seen in the table, our weighted sampling strategy for reconstructing hyperspectral images matches or improves upon HT detection performance of prior work in all cases with a 1$\%$ margin of error. This performance is achieved by reconstructing noise-reduced versions of the hyperspectral images directly by training on sample averages. While reconstructions with the DCT basis still required approximately just as many measurements as prior work to achieve its results, both learned dictionary bases allowed much fewer measurements to achieve their results. The best performance overall was achieved with the larger dictionary containing $D = 100$ bases, which yielded perfect Tro-

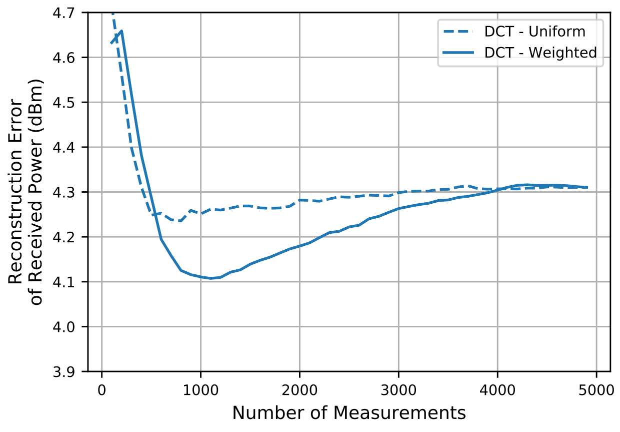
Table 5.2: HT detection performance for each reconstruction basis, compared against prior results on the AES-T1800 circuit with trigger monitoring a given number of bits.

Trigger Size (bits)	Prior Work (AUC)	DCT Basis $D = 49$ (AUC)	Dict. $D = 50$ (AUC)	Dict. $D = 100$ (AUC)
128	1.000	1.000	0.990	1.000
64	1.000	1.000	1.000	1.000
32	1.000	1.000	0.990	1.000
16	1.000	1.000	0.990	1.000
8	0.983	0.991	0.990	1.000
4	1.000	0.992	0.990	1.000
2	0.873	0.994	0.990	1.000
Sampling required vs. prior work	100% (5000)	98% (4900)	36% (1800)	10% (500)

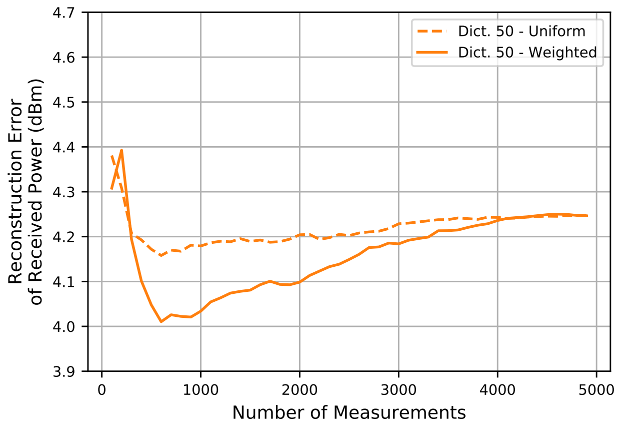
jan detection performance while requiring only 500 measurements or approximately 10% of those required in previous work. These results indicate that larger dictionaries would match performance smaller dictionaries at the least, and significantly reduce measurement requirements at the best. To robustly learn larger dictionaries would require more data to avoid simply memorizing the training data. Overall, these results demonstrate that bases closely tied to the type of measurements captured can not only improve HT detection, but also significantly reduce measurement requirements to do so. The backscattering EM hyperspectral images captured to non-destructively detection dormant hardware Trojans are well represented by compressed measurements when reconstructing with a learned dictionary of that data as expected.

5.5.2 Reconstruction Analysis

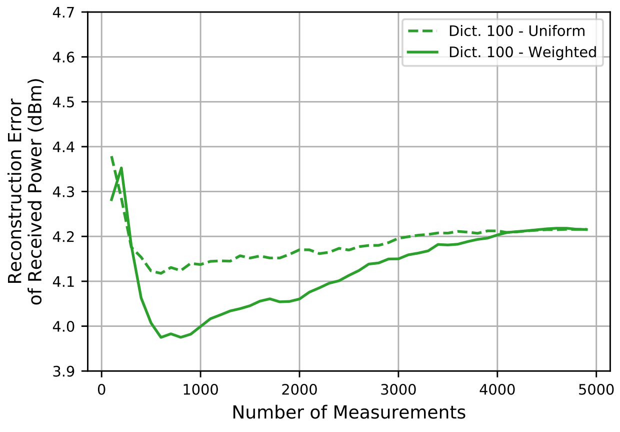
To further demonstrate the effect of our reconstruction strategy, we analyze the reconstruction accuracy for each of the bases tested and for both a uniform and our weighted sampling approaches to sampling in the compressed sensing framework. For each device we use for training, we first generate a set of reconstructed images from an initial set of 100 measurements. We then repeat the reconstruction with 100 more random measurements, and repeat



(a)



(b)



(c)

Figure 5.5: Image reconstruction error for uniform random sampling compared to our reliability-weighted sampling method for the DCT bases (a), learned dictionary with 50 bases (b), and learned dictionary with 100 bases (c).

until we capture 5000 measurements which is approximately the number of measurements used in Chapter 4. For each of these reconstructions, we compare the result with the average measurements from that device. For each of the DCT, 50-length dictionary, and 100-length dictionary we use the best value for α selected as described in Subsection 5.5.1 for the learned dictionary of 100 bases to ensure each of the bases is given equal comparison for the two sampling methods sampling methods. We reconstruct the full hyperspectral image from five unique trials of random measurements for each increment of 100 measurements.

As seen in Figure 5.5, our reliability-weighted random sampling method generally results in less reconstruction error for each of the CS bases we tested here. The difference in performance between sampling methods is greatest at moderately-low sampling rates and is confirmation of our hypothesis that weighting sampling toward unreliable measurements can more quickly approximate the hyperspectral backscattered EM side-channel signals. Reconstruction errors for both sampling methods are largest when the fewest samples are captured. Nonetheless, the uniform sampling method slightly outperforms our weighted sampling method with extremely low sampling rates. The difference in reconstruction performance between the sampling methods decreases as the number of measurements increases which can be expected as each of the random sampling methods begins to cover the entire hyperspectral space with greater probability. We also find that the lowest reconstruction error occurs at lower sampling rates for the learned dictionary bases compared to the DCT bases. This follows intuition because we can expect that our data-driven bases allow for more accurate reconstruction with fewer samples of these hyperspectral images than the data-agnostic approach using the DCT basis.

It may counter intuition that reconstruction error increases with more measurements as seen in the figure. However, we note that these results are presented when using a single reconstruction sparsity parameter α which leads to optimal HT detection performance. That α does not necessarily lead to optimal reconstruction performance. It can be expected that reconstruction with $\alpha \to 0$ would provide the optimal reconstruction performance and a monotonically non-increasing graph of reconstruction error versus number of measurements. In accordance with the Trojan detection results presented in the previous Subsection 5.5.1, these results show that the reconstruction error is lowest for the learned dictionary with 100 bases. Since we are measuring reconstruction error against the average measurements of the entire hyperspectral space, we can conclude that significantly better reconstruction error of noise-free measurements leads to better Trojan detection performance.

5.6 Conclusions

This work, submitted to [35], established a novel compressed sensing strategy for reconstructing hyperspectral images from reliability-weighted random samples. The methods presented improve hardware Trojan detection performance versus prior work and do so with up to ten times fewer measurements. That performance is achieved by biasing random sampling toward hyperspectral locations which are known to have more variation in the backscattered EM side-channel signal; meaning they are unreliable locations across different integrated circuits. We test random trials of our methods across a range of sampling rates and sparsity levels in addition to three separate reconstruction bases. We find that data-driven bases allow for much sparser sampling to achieve similar reconstruction error and HT detection performance. We also show that our weighted random sampling strategy is able to recover images with lower error than standard compressed sensing with uniform random sampling.

The performance of the compressed sensing and weighted sampling approaches developed in this work motivates further research into sparse scanning methods to reconstruct hyperspectral images of backscattering EM side-channel images. Future trials are needed to demonstrate the robustness of these methods to a variety of circuits and Trojan types in addition to manufacturing variabilities for other ICs. This work makes significant progress toward realizing high-throughput IC validation, but further work is required to implement these methods in real time. Additionally, it remains to be seen if other bases may further reduce sampling requirements or improve detection.

CHAPTER 6
RESEARCH CONTRIBUTIONS AND FUTURE WORK

6.1 Research Contributions

As designers increasingly choose to outsource their device fabrication to untrusted entities, validating hardware before sending to customers becomes crucial to ensure the safety and security of those devices. Malicious tampering with the security of integrated circuits could come in the form of counterfeit components, injection of a hardware Trojan, or other covert interference changing their expected form and function. Being able to detect these intrusions non-destructively and at high throughput is essential to ensure the security of all devices produced. This research addresses these needs by developing machine learning methods to fingerprint components based on their electromagnetic side-channel emanations or reflections and detect counterfeits or covert hardware Trojans. The contributions of this research are summarized below:

- We develop a method for device designers to non-destructively validate the identity of several types of components on already-assembled devices. Monitoring the side channel emanations of devices has been shown to uncover anomalous software activity occurring on a device. The electromagnetic side channel has been shown to leak information from these devices with much higher bandwidth than other side channels like temperature, power, or acoustic side channels. By measuring these information-rich emanations, we learn component type- and model-specific fingerprints that are used to classify unknown components as known models or uncover unseen counterfeits. We obtain robust results when identifying several component models of processor, memory, ethernet, and power management integrated circuits in two different usage scenarios. The first scenario emulates a device designer testing components on a newly-manufactured device that has just been powered on for final testing before it is deployed. The second scenario uses repeating program activity to test an actively-deployed device that is already in use. Our results demonstrate that this method is effective for a wide variety of component models and usage scenarios so that it may feasibly be employed in real world test environments.

- To detect component models robustly, we designed a novel deep learning architecture, pre-processing, and validation strategy to accommodate the high-dimensional, data-scarce scenario using electromagnetic side-channel signals. This novel convolutional architecture employs a unique combination of pooling, kernel stride, and weight dropout to heavily reduce the dimensionality of input signals while maintaining the visible and obscure spectral features that distinguish the electromagnetic emanations of different integrated circuit models. With feasible measurement times making it difficult to capture sufficient samples to train normal convolutional neural networks, this architecture is lightweight, can be trained quickly, and could be used for real-time inference for automating counterfeit detection.

- This work also defines a new model analysis tool called a feature activation map which we use to compare discriminative features learned by the deep network with conventional hand-crafted spectral features. By analyzing the intermediate representation of the inputs after passing through the convolution layers, we demonstrate how hand-crafted features correspond with model-learned features at a coarse level. We then analyze the average neuron weights in the first linear layer as compared to the activation of those intermediate outputs when passing to the first linear layer to understand which of those model-learned features are most important when classifying individual component models. This model meta analysis lends greater interpretability to a class of models which traditionally is seen as a "black box."

- We design a non-destructive measurement methodology for capturing point-scanned hyperspectral images of the backscattering electromagnetic side channel to measure circuit fingerprints. This non-traditional imaging methodology captures measurements across space and frequency by physically scanning a high resolution probe in two dimensions across the face of an integrated circuit. The method uses the probe tip to emit a carrier signal toward the board and the probe coil to measure the backscattered signal that is modulated by the device's clock frequency. At a specific location on the circuit, we measure a small bandwidth around the expected location of each of the modulated clock frequency harmonics and record the maximum power which corresponds to the frequency spike of that harmonic. By aggregated these recordings across space and a set of harmonics, we generate a hypercube of measurements that can be used to fingerprint the architecture and transistor state variations across the circuit.

- This research designs a method to capture hyperspectral images which may exhibit significant noise variability, but without needing to repeat full scans of the entire hyperspectral space. We develop an active learning and feature selection approach for capturing and comparing hyperspectral images of the backscattering EM side channel. Our active sampling approach only repeats sampling to reduce measurement error when the distribution of samples do not closely match or significantly deviate from expected scans. Then, we only compare measurements at hyperspectral locations which are known to be more reliable across multiple copies of the same integrated circuit. Comparing the sparse set of locations increases how distinguishable different circuits are by eliminating noise-ridden features that do not contribute meaningfully to the classification task.

- Additional research efforts into hyperspectral scanning of the backscattering electromagnetic side channel resulted in a novel technique for recovering these images with heavily reduced measurement cost. We design a reliability-weighted compressed sensing technique which non-uniformly randomly samples hyperspectral points before they are recovered with traditional compressed sensing optimization. Our novel technique to measure feature reliability compares the spread of measurements across different copies of the same circuit with the average spread of measurements for individual circuits. Sampling with the strategy reduces measurement costs up to ten times below previous work and reconstructs hyperspectral images more accurately and with fewer measurements than a uniform random sampling approach.

- These hyperspectral scanning approaches lead to state-of-the-art non-destructive detection of hardware Trojans. Where prior work struggled to detect Trojans less than about 0.35% of the size of the of the circuit they infect, our research achieved robust detection for Trojans down to 0.03% of the circuit size. While we make a trade-off of increased sampling time, our methods reduce the detectable size of Trojans by about 14 times. The unique sampling methods we develop make significant progress toward making these hardware Trojan detection techniques operate at high throughput.

6.2 Future Research Directions

While the learning and measurement methods developed over the course of this research have shown significant improvements over existing methods, much work still remains for these to be implemented in real-world scenarios. Our component identification and counterfeit detection methods demonstrate good performance for a variety of test scenarios and component types. First, we use a simple anomaly detection procedure based solely off of the CNN's output activations, whereas the state-of-the-art in that domain takes cues from all layers in the CNN and could be significantly improved. We do not claim to have made a comprehensive architecture search for the deep learning model and it is likely that additional architecture tweaks could improve performance for each test case. Individual model architectures tuned to each of the component types or testing scenarios could better match the peculiarities of each problem, since the features that distinguish processors are not necessarily the same as those that distinguish power management integrated circuits. As with any deep learning problem, larger source datasets could further demonstrate the robustness of our methods and data augmentation could be one way of making strides in that direction.

The strong performance of the methods developed for detecting dormant hardware Trojans work motivates further investigation into the problem. Feature selection methods and hyperspectral scanning of the backscattering EM side channel for hardware Trojan detection enabled detection of much smaller Trojans than had previously been achieved. Future work is still needed to demonstrate the success of these methods for other Trojan and trigger varieties that do not scale in size like the internally-activated, condition-based HTs we tested here. Additional study would be useful to analyze the robustness to manufacturing variabilities for other ICs, especially for ASICs or those with higher transistor density. Further maturing the active sampling methods to reduce hyperspectral sampling time for higher testing throughput is an open area of research. While the research presented in this thesis assumes the possession of at least one uninfected device, an extension of this work may demonstrate these methods without that assumption by using circuit simulations, one class classification, anomaly detection, or other methods.

Our initial study into compressed sensing techniques has yielded great results in reducing measurement time for hyperspectral scans. While we only scanned a small 7×7 grid of locations on the chip, it is likely that sampling of a greater area would lend itself better to the image reconstruction methods we test here. It is even possible that more dense sampling or super-resolution reconstructions could be achieved with compressed sensing or deep learning models. It remains to be seen whether sampling larger areas of chips or scanning at higher density would be worth the trade-off in scanning time for more robust or smaller Trojan detection. These tests were performed on a relatively small set of device copies and simply spending the time to measure many more devices could yield much more confident statistics about the reliability of our methods. Further validation is necessary to determine whether our most fine-grained Trojan detection results can be achieved at scale and with what confidence those classifications can be made. An ideal non-destructive Trojan detection strategy would be entirely automated and performed at high throughput; both of which are left as an exercise for the reader.

In addition to the aforementioned extensions to our existing work, a more general open problem remains to understand how backscattered electromagnetic side-channel signals correspond to the underlying physical characteristics of an integrated circuit. Previous work and ours has shown that small differences in the underlying trace architecture of an IC will result in changes to the backscattered signal. While that fact has allowed us to uncover the presence of anomalies, a more general understanding of the relationship between specific architecture changes and backscattered signal could be very useful. With a direct relationship between trace architecture and expected backscattered signals, one could conceivably reverse engineer not only *if*, but *what* changes had been made to a circuit solely from analyzing the side channel. Having a thorough correspondence between netlist layout, high resolution physical layout (from destructive imagery methods), and side-channel emanations or reflections could improve the feasibility of non-destructively characterizing IC modifications.

REFERENCES

[1] N. Kae-Nune and S. Pesseguier, “Qualification and testing process to implement anti-counterfeiting technologies into ic packages,” in *Proc. 2013 Des. Autom. Test Eur. Conf. Exhib. (DATE)*, 2013, pp. 1131–1136.

[2] M. Pecht and S. Tiku, “Bogus: Electronic manufacturing and consumers confront a rising tide of counterfeit electronics,” *IEEE Spectrum*, vol. 43, no. 5, pp. 37–46, 2006.

[3] E. Menzel and E. Kubalek, “Fundamentals of electron beam testing of integrated circuits,” *Scanning*, vol. 5, no. 3, pp. 103–122, 1983.

[4] Y. Jin, D. Maliuk, and Y. Makris, “Hardware trojan detection in analog/rf integrated circuits,” 2016.

[5] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder, “Acoustic side-channel attacks on printers,” in *Proc. 19th USENIX Conf. Secur.*, 2010, p. 20.

[6] M. Hutter and J.-M. Schmidt, “The temperature side-channel and heating fault attacks,” in *Smart Card Research and Advanced Applications: 12th Int. Conf. (CARDIS)*, vol. 8419, Nov. 2013.

[7] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Advances in Cryptology (CRYPTO)*, vol. 1666, Dec. 1999.

[8] W. van Eck, “Electromagnetic radiation from video display units: An eavesdropping risk?” *Computers & Security*, vol. 4, no. 4, pp. 269–286, 1985.

[9] R. Callan, A. Zajić, and M. Prvulovic, “A practical methodology for measure the side-channel signal available to the attacker for instruction level events,” *IEEE MICRO*, vol. 14, pp. 1–12, 2013.

[10] D. Agrawal and B. Archambeult, “The EM side-channel(s),” *Proc. Crypto. HW and Emb. Sys. (CHES)*, pp. 29–45, 2002.

[11] B. B. Yilmaz, A. Zajić, and M. Prvulovic, “Capacity of EM side channel created by instruction executions in a processor,” *Processings of IEEE IEMCON*, pp. 1–5, 2019.

[12] M. Kuhn, “Compromising emanations: Eavesdropping risks of computer displays,” Computer Laboratory, University of Cambridge, Tech. Rep., Apr. 2004.

[13] M. Alam *et al.*, "One&done: A single-decryption em-based attack on openssl's constant-time blinded rsa," in *Proc. 27th USENIX Secur. Symp.*, Aug. 2018.

[14] X. Dong *et al.*, "Detection and identification of vehicles based on their unintended electromagnetic emissions," *IEEE Trans. Electromagn. Compat.*, vol. 48, no. 4, pp. 752–759, 2006.

[15] H. Göksu, D. Wunsch, X. Dong, A. Kökce, and D. Beetner, "Detection and identification of vehicles based on their spark-free unintended electromagnetic emissions," *IEEE Trans. Electromagn. Compat.*, vol. 60, pp. 1594–1597, Oct. 2018.

[16] M. M. Ahmed *et al.*, "Authentication of microcontroller board using non-invasive em emission technique," in *2018 IEEE 3rd Int. Verif. Secur. Workshop (IVSW)*, 2018, pp. 25–30.

[17] B. B. Yilmaz, E. Mert Ugurlu, A. Zajić, and M. Prvulovic, "Cell-phone classification: A convolutional neural network approach exploiting electromagnetic emanations," in *Proc. 2020 IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 2862–2866.

[18] W. E. Cobb, "Exploitation of unintentional information leakage from integrated circuits," Ph.D. dissertation, Air Force Institute of Technology, 2017.

[19] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 15–19.

[20] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[21] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proc. IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[22] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, May 2016.

[23] M. Tehranipoor *et al.*, "Trustworthy hardware: Trojan detection and design-for-trust challenges," *Computer*, vol. 44, no. 7, pp. 66–74, 2011.

[24] S. Mal-Sarkar, R. Karam, S. Narasimhan, A. Ghosh, A. Krishna, and S. Bhunia, "Design and validation for fpga trust under hardware trojan attacks," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 186–198, 2016.

[25] L. N. Nguyen, C.-L. Cheng, M. Prvulovic, and A. Zajić, "Creating a backscattering side channel to enable detection of dormant hardware trojans," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1561–1574, 2019.

[26] J. Landt, "The history of rfid," *IEEE Potentials*, vol. 24, no. 4, pp. 8–11, 2005.

[27] C. Boyer and S. Roy, "— invited paper — backscatter communication and rfid: Coding, energy, and mimo analysis," *IEEE Transactions on Communications*, vol. 62, no. 3, pp. 770–785, 2014.

[28] H. Dogan, D. Forte, and M. M. Tehranipoor, "Aging analysis for recycled fpga detection," in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2014, pp. 171–176.

[29] E. J. Jorgensen, A. Kacmarcik, M. Prvulovic, and A. Zajić, "Novel feature selection for non-destructive detection of hardware trojans using hyperspectral scanning," *Journal of Hardware and Systems Security*, vol. 6, to appear, 2022.

[30] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[31] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[32] M. Gehm, R. John, D. Brady, R. Willett, and T. Schulz, "Single-shot compressive spectral imaging with a dual-disperser architecture," *Optics express*, vol. 15, pp. 14 013–27, Nov. 2007.

[33] A. Wagadarikar, R. John, R. Willett, and D. Brady, "Single disperser design for coded aperture snapshot spectral imaging," *Applied optics*, vol. 47, B44–51, May 2008.

[34] E. Jorgensen, F. Werner, M. Prvulovic, and A. Zajic, "Deep learning classification of motherboard components by leveraging em side-channel signals," *J. Hardw. Syst. Secur.*, vol. 5, Jun. 2021.

[35] E. J. Jorgensen, A. Kacmarcik, M. Prvulovic, and A. Zajić, "Hyperspectral image recovery via reliability-weighted compressed sensing for hardware trojan detection," in press.

[36] U. Guin, D. Dimase, and M. Tehranipoor, "Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead," *J. Electron. Test.: Theory Appl.*, vol. 30, pp. 9–23, Feb. 2014.

[37] M. Tehranipoor and F. Koushanfar, “A survey of hardware trojan taxonomy and detection,” *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[38] Sparkfun, *Dear on semiconductor*, https://www.sparkfun.com/news/384 (2022/6/26), Jun. 2010.

[39] F. T. Werner, B. B. Yilmaz, M. Prvulovic, and A. Zajić, “Leveraging em side-channels for recognizing components on a motherboard,” *IEEE Transactions on Electromagnetic Compatibility*, pp. 1–14, 2020.

[40] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, “Hardware trojan: Threats and emerging solutions,” in *2009 IEEE International High Level Design Validation and Test Workshop*, 2009, pp. 166–171.

[41] V. Venugopalan and C. D. Patterson, “Surveying the hardware trojan threat landscape for the internet-of-things,” *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 131–141, 2018.

[42] Y. Jin, N. Kupp, and Y. Makris, “Experiences in hardware trojan design and implementation,” in *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2009, pp. 50–57.

[43] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, “Trustworthy hardware: Identifying and classifying hardware trojans,” *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[44] J. Zhang, F. Yuan, and Q. Xu, “Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’14, Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 153–166, ISBN: 9781450329576.

[45] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, “Benchmarking of hardware trojans and maliciously affected circuits,” *J. Hardw. Syst. Secur.*, vol. 1, Mar. 2017.

[46] J. Rajendran, O. Sinanoglu, and R. Karri, “Regaining trust in vlsi design: Design-for-trust techniques,” *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1266–1282, 2014.

[47] P. Rohatgi, “Electromagnetic attacks and countermeasures,” in Nov. 2008, pp. 407–430, ISBN: 978-0-387-71816-3.

[48] T. Komolafe, W. Tian, G. T. Purdy, M. Albakri, P. Tarazaga, and J. Camelio, “Repeatable part authentication using impedance based analysis for side-channel monitoring,” *Journal of Manufacturing Systems*, vol. 51, pp. 42–51, 2019.

[49] P. Nikitin and K. Rao, "Theory and measurement of backscattering from rfid tags," *IEEE Antennas and Propagation Magazine*, vol. 48, no. 6, pp. 212–218, 2006.

[50] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital integrated circuits*. Prentice hall Englewood Cliffs, 2002, vol. 2.

[51] A. F. Goetz, "Three decades of hyperspectral remote sensing of the earth: A personal view," *Remote Sensing of Environment*, vol. 113, S5–S16, 2009.

[52] O. Carrasco, R. B. Gomez, A. Chainani, and W. E. Roper, "Hyperspectral imaging applied to medical diagnoses and food safety," in *Geo-Spatial and Temporal Image and Data Exploitation III*, International Society for Optics and Photonics, vol. 5097, 2003, pp. 215–221.

[53] G. Lu and B. Fei, "Medical hyperspectral imaging: a review," *Journal of Biomedical Optics*, vol. 19, no. 1, pp. 1–24, 2014.

[54] R. R. Carvalho, J. A. Coelho, J. M. Santos, F. W. Aquino, R. L. Carneiro, and E. R. Pereira-Filho, "Laser-induced breakdown spectroscopy (LIBS) combined with hyperspectral imaging for the evaluation of printed circuit board composition," *Talanta*, vol. 134, pp. 278–283, 2015.

[55] R. Palmieri, G. Bonifazi, and S. Serranti, "Recycling-oriented characterization of plastic frames and printed circuit boards from mobile phones by electronic and chemical imaging," *Waste Management*, vol. 34, no. 11, pp. 2120–2130, 2014.

[56] H. Grahn and P. Geladi, *Techniques and applications of hyperspectral image analysis*. John Wiley & Sons, 2007.

[57] N. A. Hagen and M. W. Kudenov, "Review of snapshot spectral imaging technologies," *Optical Engineering*, vol. 52, no. 9, pp. 1–23, 2013.

[58] S. Adibelli, P. Juyal, L. N. Nguyen, M. Prvulovic, and A. Zajić, "Near-field backscattering-based sensing for hardware trojan detection," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 12, pp. 8082–8090, 2020.

[59] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[60] R. H. Chan, T. F. Chan, L. Shen, and Z. Shen, "Wavelet algorithms for high-resolution image reconstruction," *SIAM Journal on Scientific Computing*, vol. 24, no. 4, pp. 1408–1432, 2003.

[61] H. Rauhut, K. Schnass, and P. Vandergheynst, "Compressed sensing and redundant dictionaries," *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2210–2219, 2008.

[62] E. J. Candes, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, 2011.

[63] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[64] R. J. Tibshirani, "The lasso problem and uniqueness," *Electronic Journal of statistics*, vol. 7, pp. 1456–1490, 2013.

[65] R. Taylor, *Compressed Sensing in Python*, Available at http://www.pyrunner.com/weblog/2016/05/26/compressed-sensing-python/ (2022/6/29), 2016.

[66] F. Werner, D. A. Chu, A. R. Djordjević, D. I. Olćan, M. Prvulovic, and A. Zajić, "A method for efficient localization of magnetic field sources excited by execution of instructions in a processor," *IEEE Transactions on Electromagnetic Compatibility*, vol. 60, no. 3, pp. 613–622, 2018.

[67] R. Rubino, "Wireless device identification from a phase noise prospective," M.S. thesis, University of Padova, Italy, 2010.

[68] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 315–323.

[69] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[70] Olimex, *A10-olinuxino-lime*, Available at https://www.olimex.com/Products/OLinuXino/A10/A10-OLinuXino-LIME-n4GB/open-source-hardware (2020/05/08).

[71] Olimex, *A13-olinuxino*, Available at https://www.olimex.com/Products/OLinuXino/A13/A13-OLinuXino/open-source-hardware (2020/05/08).

[72] Olimex, *A13-olinuxino-micro*, Available at https://www.olimex.com/Products/OLinuXino/A13/A13-OLinuXino-MICRO/open-source-hardware (2020/05/08).

[73] Olimex, *A20-olinuxino-lime*, Available at https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXino-LIME/open-source-hardware (2020/05/08).

[74] Olimex, *A20-olinuxino-lime2*, Available at https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXino-LIME2/open-source-hardware (2020/05/08).

[75] Olimex, *A20-olinuxino-micro*, Available at https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXino-MICRO/open-source-hardware (2020/05/08).

[76] Olimex, *A20-olinuxino*, Available at https://www.olimex.com/Products/OLinuXino/A33/A33-OLinuXino/open-source-hardware (2020/05/08).

[77] Riscure, *Em probe station*, Available at https://getquote.riscure.com/en/quote/2101064/em-probe-station.htm (2020/05/08).

[78] K. Technologies, *M9391a pxie vector signal analyzer: 6 ghz*, Available at https://www.keysight.com/en/pd-2317933-pn-M9391A/pxie-vector-signal-analyzer-1-mhz-to-3-ghz-or-6-ghz?&cc=US&lc=eng (2020/05/08).

[79] M. Prvulovic, A. Zajić, R. L. Callan, and C. J. Wang, "A method for finding frequency-modulated and amplitude-modulated electromagnetic emanations in computer systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, no. 1, pp. 34–42, 2017.

[80] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018, pp. 7167–7177.

[81] H. A. Khan, N. Sehatbakhsh, L. N. Nguyen, M. Prvulovic, and A. Zajić, "Malware detection in embedded systems using neural network model for electromagnetic side-channel signals," *Journal of Hardware and Systems Security*, vol. 3, no. 4, pp. 305–318, 2019.

[82] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.

[83] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[84] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, *Class-balanced loss based on effective number of samples*, 2019. arXiv: 1901.05555 [cs.CV].

[85] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware trojans using power supply transient signals," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 3–7.

[86] C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification*. Springer Science & Business Media, 2003, vol. 1.

[87] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware trojan detection," in *2015 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, 2015, pp. 246–251.

[88] M. Cerna and A. F. Harvey, "The fundamentals of fft-based signal analysis and measurement," Application Note 041, National Instruments, Tech. Rep., 2000.

[89] Intel, *Glossary*, Available at https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/glosslist.htm# (2022/04/28).

[90] terasIC, *De0-cv board*, Available at https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=163&No=921 (2022/02/16).

[91] Intel, *Cyclone v device overview*, Available at https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_51001.pdf (2021/05/27).

[92] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *Proc. 31st IEEE Int. Conf. Comput. Des.*, 2013, pp. 471–474.

[93] Zaber, *X-lsq150b specifications*, Available at https://www.zaber.com/products/linear-stages/X-LSQ/specs?part=X-LSQ150B (2022/05/2).

[94] R. Wilson, H. Lu, M. Zhu, D. Forte, and D. L. Woodard, "Refics: Assimilating data-driven paradigms into reverse engineering and hardware assurance on integrated circuits," *IEEE Access*, vol. 9, pp. 131 955–131 976, 2021.

[95] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.