ABSTRACT

Title of Dissertation: Intellectual Property Protection:
From Integrated Circuits to Machine Learning Models

Omid Aramoon, Doctor of Philosophy, 2022

Dissertation Directed by: Professor Gang Qu
Department of Electrical and Computer Engineering

The increasing popularity of intellectual property (IP) based design in the semiconductor and artificial intelligence (AI) industry has created a growing market for silicon and machine learning (ML) IPs. The emerging IP market in both sectors has facilitated the exchange of designs and ideas among entities, which in turn has helped speed up innovations, lower R&D costs, and shorten the time-to-market for new products. Nonetheless, two major concerns have been raised in the IP market that may overshadow these benefits and discourage suppliers (IP vendors) and consumers (IP buyers) from entering the IP market. First, there is the issue of IP infringements, which negatively impact IP vendors. Given that IPs can easily be copied and distributed, sharing them with other entities in a market environment increases the risk of IP theft and copyright violations. Such infringements would erode the profit margins of IP vendors and discourage them from investing in IP development. The second issue pertains to IP buyers, who are primarily concerned about how using third-party IPs might impact the safety and security (S&S) of their systems. Many real-world applications require designers to provide S&S assurance for their

products. However, this becomes challenging for systems that make use of third-party IPs since IP buyers often lack the necessary knowledge about the core design features of commercial IPs to devise effective S&S measures.

In this thesis, our goal is to develop technical solutions to address these two concerns in order to promote participation in the semiconductor and AI IP markets and thereby stimulate faster growth in both sectors.

The first part of this thesis is dedicated to addressing vendors' concerns regarding IP infringements by proposing IP watermarking and fingerprinting solutions. Protecting IPs through legal means is passive and ineffective unless forensic means such as IP watermarking and IP fingerprinting are available to assist vendors in establishing ownership over pirated IPs and identifying the source of infringement. In this direction, we make four contributions: **(1)** Our first contribution is a dynamic watermarking scheme for silicon IPs that relies on the multi-functionality of polymorphic gates to hide ownership information in circuits. With the proposed watermarking method, the circuit functions as expected at normal operating temperature; however, when the circuit is heated, the hidden behavior of polymorphic gates is activated and the circuit's functionality changes to reveal the watermark. Experiment results demonstrate that our scheme can embed large multi-bit signatures while incurring low overhead in terms of performance, area, and power consumption. **(2)** The second contribution is a black-box watermarking method for ML IPs, particularly deep neural network (DNN) classifiers, which we call GradSigns. The proposed scheme embeds the ownership information as a set of stego-constraints on the gradients of model components. Our experiments suggest that GradSigns is extremely robust to counter-watermark attacks and is capable of embedding large multi-bit signatures without sacrificing the performance of the model, two properties that were lacking in the prior art. **(3)** The third

contribution is a fingerprinting scheme for silicon IPs that replaces standard cells holding "Satisfiability Don't Care" (SDC) conditions with signal-controlled polymorphic gates. With the proposed approach, each copy of the IP and its corresponding buyer can be identified based on the configuration of the polymorphic gates, i.e. the IP fingerprint. This attribute can help vendors trace the source of IP piracy, if needed. Experiments demonstrate that our method can provide sufficiently strong fingerprints with about half the overhead of similar methods. **(4)** The fourth and final contribution in this direction is a fingerprinting technique where the standard testing infrastructure in system-on-chips (SoCs) design is repurposed to create unique fingerprints. To this end, we adopt the reconfigurable scan network (RSN) in SoCs and develop a fingerprinting protocol that configures a unique RSN for each sold copy by utilizing the different connection styles between scan cells. Experiments show that the proposed method is capable of creating a large number of distinct fingerprints while incurring little overhead.

The second part of this thesis is dedicated to addressing IP buyers' concerns regarding the security and safety risks of using third-party IPs, with an emphasis on ML IPs. Commercial models are primarily marketed as black box oracles to reduce the risk of IP infringements. However, having little knowledge about the design details of commercial models can complicate IP buyers' efforts in addressing various S&S threats that may arise in real-world applications of ML. In this thesis, we specifically discuss two of such concerns, namely (a) inaccuracy and overconfidence of DNN classifiers in the presence of anomalous inputs, and (b) the threat from model tampering (or model integrity) attacks, and explain why existing countermeasures aren't applicable to black-box commercial DNNs. The following two contributions are made to address this shortcoming: **(1)** Our first contribution is a tamper detection technique, called AID (Attesting the Integrity of DNNs). The proposed method generates a set of input-output test cases that can

reveal whether a model has been tampered with. AID does not require access to parameters of models and thus is compatible with black-box commercial models. Experimental results show that AID is highly effective and reliable, in that, with at most four test cases, AID is able to detect eight representative integrity attacks with zero false-positive. **(2)** The second contribution in this direction is Pad-Lock, a Power side-channel-based Anomaly Detection framework for black-box DNN classifiers. The proposed method uses the power side-channel information during DNN inference operation as a proxy for the model's inner computation and discovers patterns that can be used to detect anomalous inputs such as adversarial and out-of-distribution samples based on this information. Upon preliminary examination, Pad-Lock appears to be a practical and effective framework for detecting anomalies in black-box commercial DNNs.

In summary, the methods presented in this dissertation fortify the protection of semiconductor and ML IPs against IP infringement activities and assist IP buyers in ensuring the safety and security of systems containing commercial IPs. We believe these technical solutions constitute a major step toward addressing concerns raised in the semiconductor and AI IP markets, and will ultimately encourage more entities to participate in both markets.

Intellectual Property Protection: From Integrated Circuits to Machine Learning Models

by

Omid Aramoon

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Gang Qu, Chair/Advisor
Professor Behtash Babadi
Professor Joseph Jaja
Professor Dinesh Manocha
Professor William Regli, Dean's Representative

Dedication

To my parents

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my research advisor, Prof. Gang Qu, whose insightful guidance helped me through my Ph.D. journey. There are no words to describe just how accommodating and understanding he has been to me throughout my graduate studies. He was kind enough to let me explore research areas that I was passionate about, even when they weren't quite in line with other projects in our research group. If it wasn't for his unwavering support and encouragement, this doctoral dissertation would not have materialized. I am truly honored and always grateful to be one of his students.

My sincere gratitude goes out to our collaborator, Dr. Pin-Yu Chen, who is an expert in adversarial machine learning. During the course of my Ph.D. program, he devoted a significant amount of his time to assisting me in my research, and I have learned a great deal from him.

Next, I would like to thank Prof. Behtash Babadi, Prof. Joseph JaJa, Prof. Dinesh Manocha, and Prof. William Regli for serving on my dissertation committee. I am grateful for their constructive feedback and support in the preparation of my dissertation.

Additionally, I would like to thank all my colleagues in the Maryland Embedded System and Hardware Security lab for their help and support, especially Chen Xi, Qian Xu, Zhaojun Lu, Mingze Gao, Qian Wang, and Md Tanvir Arafin. Working with these talented and smart individuals was a pleasure.

My Ph.D. research was sponsored by National Science Foundation award "EAGER: Designing

Novel Polymorphic Gates as Hardware Security Primitives" under award number 1745466, by the Air Force Office of Scientific Research under the grant "MURI: Developing of Universal Security Theory for Evaluation and Design of Nanoscale Devices", and by the Defense Advanced Research Projects Agency's "Automatic Implementation of Secure Silicon program (AISS)" under agreement number HR0011-20-F-0045.

Last but not least, I want to thank my parents, brothers, and my amazing girlfriend for their unconditional love throughout the thick and thin of this journey.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

EDA	Electronic Design Automation
IC	Integrated Circuit
IP	Intellectual Property
SIP	Silicon Intellectual Property
AI	Artificial Intelligence
ML	Machine Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
DRL	Deep Reinforcement Learning
GPU	Graphic Processing Units
HDL	Hardware Description Language
MLaaS	Machine Learning as a Service
API	Application Programming Interface
3PIP	Third-party IP
S&S	Safety and Security
DNN	Deep Neural Network
SDC	Satisfiability Don't Care
SoC	System-on-Chip
RSN	Reconfigurable Scan Network
FSM	Finite State Machine
DfT	Design-for-Testability
STG	State Transition Graph
PNG	Pseudo-random Number Generator
DFF	D Flip-Flop
SFF	Scannable Flip-Flop
SI	Scan Input
SD	Scan Data
SE	Scan Enable
TC	Test Control
SfT	Synthesis-for-Testability
GAN	Generative Adversarial Network
NAS	Neural Architecture Search
GA	Genetic Algorithm
DIV	Differentiating Input Value

RNG	Random Number Generator
SLP	Single Layer Perceptron
BER	Bit Error Rate
YTF	You Tube Faces
BESR	Bit Embedding Success Rate
SP Attack	Score Perturbation Attack
CSU	Capture-Shift-Update
SIB	Segment Insertion Bit
ID	In-Distribution
OOD	Out-Of-Distribution
AE	Adversarial Example
ECC	Error Correction Code
AID	Attesting the Integrity of DNNs
RS	Reactivity Score
VC	Validation Coverage
TBT	Targeted Trojan Bit
GDA	Gradient Descent Attack
SCA	Side-Channel Attack
MLP	Multi Layer Perceptron
TDC	Time-to-Digital Converter
PDN	Power Distribution Network
RO	Ring Oscillator
PSC2	Power Side-Channel Collection
AD	Anomaly Detection
BRAM	Block Random Access Memory
ONNX	Open Neural Network Exchange
FIFO	First-In-First-Out
UART	Universal Asynchronous Receiver Transmitter
MSE	Mean-Squared-Error
ADAM	Adaptive Moment Estimation
TRP	True Positive Rate
TNR	True Negative Rate
PGD	Projected Gradient Descent
FGSM	Fast Gradient Sign Method
FL	Federated Learning
SecAgg	Secure Aggregation
SGC	Stochastic Gradient Descent

CWM	Coordinate-Wise Median
TM	Trimmed Mean
NB	Norm Bounding
DP	Differential Privacy
Meta-FL	Meta Federated Learning

Chapter 1: Introduction

1.1 Intellectual Property Based Design in Semiconductor and AI Industries

The exponential increase in the integration density of semiconductor technology has continuously outpaced the design productivity offered by Electronic Design Automation (EDA) tools, creating a widening gap between silicon's capacity and what can be designed on it. In other words, the number of transistors available in Integrated Circuit (IC) chips is increasing much more rapidly than any design team's ability to fully utilize them. Moreover, with the market's increasing demand for more functionalities, the reducing life-cycle and the tightening time-to-market window of electronic devices, it is no longer possible to design complex systems from scratch. To shorten design cycle times and boost design productivity, the *intellectual property* (IP) *reuse* (or *IP-based*) paradigm has been proposed and been found to be successful. In this methodology, highly sophisticated systems are partitioned into small design blocks with well-defined functionality that can be reused across different systems. These re-usable design blocks, which are referred to as *silicon IP* (SIP) cores (or blocks), are often purchased from third-party providers and integrated into the systems to be designed.

Perhaps inspired by the IP-reuse paradigm, the Artificial Intelligence (AI) industry has also begun adopting a similar methodology to tackle the increasing complexity in the design of AI solutions. Today's AI solutions are highly sophisticated artifacts comprised of a variety

of Machine Learning (ML) based components such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Deep Reinforcement Learning (DRL) models, etc. Take the autonomous systems in self-driving cars as an example. The modular pipeline of an AI-driven self-driving car is composed of several ML models responsible for performing advanced tasks such as environmental perception and localization, high-level path planning, behavioral arbitration, and driving motion controllers [11]. Engineering a learning-based solution for any one of these tasks involves the costly process of training numerous models with different architectures and hyperparameters to find a model with satisfactory performance. Depending on the complexity of these models and their targeted task, each of these rounds of trial-and-error training can take several days or even weeks to complete on cutting-edge computing platforms such as Graphics processing units (GPUs). In addition, gathering very large datasets, which is often necessary for training ML models, especially those based on deep learning, can be very expensive and time-consuming. Furthermore, the current shortage of qualified AI professionals has made it challenging to hire an AI team to work on such advanced problems [12], adding to the design costs of AI solutions. In reality, designing and training even just one production-ready ML model is often too expensive for most companies, rendering the engineering cost of a complex pipeline of models such as those in self-driving cars beyond the reach of any company alone. With the scale of AI tasks increasing, it is no longer possible to design complex systems from scratch and AI engineers have no choice but to rely on proprietary machine learning models - Machine Learning IPs - from third-party providers to minimize their infrastructure costs. As a result, *model-reuse* (or *IP-based*) design is starting to become a common practice in the AI industry, and a new market is emerging for ML IPs.

Model-reuse and IP-reuse methodologies are close parallels; both are proposed to minimize

the effort and costs involved in designing complex systems, and both are based on the concepts of information sharing and integration. In the context of this analogy, the IP market in the semiconductor and that in the AI industry have a lot in common, and the reuse-based methodology in both markets presents similar benefits and drawbacks.

1.2 Semiconductor IP Market

IP-reuse in chip design came into practice in the early 1990s and ever since it has fueled the rapid innovation in electronics by allowing system designers to focus their time and efforts on the design of critical blocks that would result in their competitive advantage in the market. Furthermore, it has lowered the overall system development costs, enabling smaller firms with limited resources to develop highly complex systems, thus creating a more competitive market.

Over the past decades, the increasing popularity of IP-reuse methodology has driven companies to jump on the IP business bandwagon, resulting in a booming market for IP cores. According to Research and Markets [13], the global semiconductor IP market is expected to grow at an annual rate of 7.5% from 2019 to 2026 and will reach 8.81 billion dollars by 2026. The major contributors to this growing market are the design houses (IP vendors) that invest in developing new, optimized IP blocks and make profits by licensing them to system designers (IP buyers).

The semiconductor market offers a large selection of IPs at various abstract levels providing designers with varying degrees of customization and portability. Silicon IP cores are typically licensed and delivered as soft, firm, or hard IPs. Hard IPs are delivered as a plug-and-play component in the form of a physical layout optimized for specific process technology. This form of IPs offers the best performance in terms of area, delay, and power consumption. However,

they can not be customized nor can be ported to another process technology. On the other hand, soft IPs, which are often delivered in the form of synthesizable hardware description language (HDL) code, are technology-independent and can easily be adapted to meet the requirements of any system. Nevertheless, there is a downside to soft IPs, in that, their performance is less predictable as it heavily depends on the optimization skills of the system designer and the process technology being used. Firm IP cores (also known as semi-hard IP cores) are shipped in the form of a gate-level netlist, an intermediate form between the soft and hard IPs. Compared to hard IPs, firm IPs tend to have inferior performance but offer better portability. In comparison to soft IPs, their portability is limited, but their performance is more predictable.

1.3 Artificial Intelligence IP Market

The AI market includes two types of products, building-oriented, and sale-oriented services. In the latter category, customers are provided with the necessary infrastructure, such as computing power, dataset, and expertise, to work on their own machine learning problems, whereas sales-oriented services provide customers with access to a wide range of pre-built models whose specifications match their needs. In this thesis, we focus on the second category and use the term AI marketplace (or AI IP market) to refer to a market for the exchange of AI models between different parties. In the AI IP market, there are two major players: the model vendor (or IP vendor) who is the entity that sells proprietary ML models, and the system designer (or IP buyer) who is the entity that purchases ML models and integrates them in the system to be designed.

Sharing pre-trained models has long been a common practice in academia to promote research reproducibility, but doing so for financial gain is a rather new development in the

industry. In the AI marketplace, models are often commercialized as black-box oracles [14, 15] following two prevalent business models namely *on-cloud* and *on-premise*. In the former scenario, which is also referred to as *Machine learning as a Service (MLaaS)* [16], vendors deploy their products on cloud machines offered by third-party companies such as Google, Amazon, and Microsoft and monetize their models through a paid application programming interface (API) access. In the on-premise model, models are commercialized similar to software and mobile apps that are to be installed on customers' premises. In this scenario, just like the on-cloud approach, models are only accessible via prediction APIs, and customers need to purchase a license to use them. The on-premise business model is mainly adopted for customers who require reliable real-time access to the purchased models or plan to use the model for processing sensitive data such as health care and financial information. In both on-cloud and on-premise models, system designers can integrate the provided APIs into their design and take advantage of the predictive capabilities of the licensed model. However, there is a downside to these business models: they offer no flexibility to systems designers to customize the models to meet their needs. It is often the case that the learning task of a customer does not completely match the original task for which the model was trained and therefore, customers need to tune and adapt the licensed models before integrating them into systems. That is why the trend toward white-box model sharing appears to be accelerating.

1.4 Challenges in the Semiconductor and AI IP Markets

Design sharing has numerous advantages. It can reduce engineering costs for complex systems and enable smaller firms with limited resources to develop competitive products. This

would increase market competition, speed up innovation, and prevent giant corporations from monopolizing the market. Additionally, design sharing can shorten design cycle times, helping system designers to stay current with market demands and keep end-users happy. The discussion so far has primarily focused on the advantages of IP sharing, when in reality, there are a number of concerns that may detract from its benefits and even discourage IP vendors and IP buyers from participating in the market. In this subsection, we discuss some of the concerns over IP transactions in the IP market.

1.4.1 IP Infringements

From IP vendors' point of view, IP infringements are arguably the most serious concern to the reliable and secure commercialization of IPs over the market. The subject of IP infringement covers a broad range of problems two of which, namely IP theft and copyright violation, are the focus of this thesis. IP theft is the act of any illegal use or unauthorized integration of an IP into a system. In both semiconductor and AI IP markets, legitimate buyers may attempt to reuse a purchased design in unlicensed systems without asking for permission and paying the additional licensing fees. This would eat into the profit margin of IP vendors and drive them out of business. Copyright violation is the act of illegal re-sell or distribution of an IP over the market. In both semiconductor and AI IP markets, a licensed buyer may re-sell the IP to other entities or may slightly modify the purchased design and redistribute it over the market under a different brand and potentially at a cheaper price to take the market share from the original IP vendor.

Developing both semiconductor and AI IPs involves substantial time, effort, and monetary investment. However, sharing designs with other entities in the market environment makes

commercial IPs more vulnerable than ever to IP infringement activities. Securing IPs through legal means such as copyright, patent, and trade secret agreements, although effective, cannot suppress IP infringements alone. This is because IP infringements are hard to detect due to hidden manifestations of IPs in complex systems and are even harder to prove in court, especially if the IP does not stand out in terms of novelty and uniqueness of its idea and design compared to the prior art. Consequently, corporations are extremely cautious in suing IP infringements since they do not want to be involved in lengthy and costly legal battles with little guarantee of success.

One reason that IP infringements are hard to detect and prove is that IP development traditionally focuses on optimizing the IP's performance and reducing its design costs, and little attention is paid to tracing and detecting IPs when they are deployed in the application fields or integrated into more complex systems. The IP market in both the semiconductor and AI industries is in dire need of a secure transaction environment, through which IP vendors can be assured that their designs are protected from IP infringements. For that, technical methods are required to detect the occurrence of IP infringements, provide concrete proof of ownership, and offer convincing evidence as to the identity of the pirate.

1.4.2 System Safety and Security with Third-Party IPs

IP buyers have their own share of concerns regarding the purchase and integration of third-party IPs (3PIPs) into their products. These concerns are related to the *safety and security* (S&S) implications of using 3PIPs. For many real-world applications, it is imperative for system designers to provide S&S assurance for their products. Nevertheless, this may prove challenging when their products contain IPs licensed from external providers because the design process of

3PIPs is not transparent to system designers, and they are unlikely to possess sufficient knowledge of the IP to devise an effective S&S measure. In this regard, it is vital that IP vendors deliver S&S assurance collateral for their technologies. This can certainly help protect systems against threats that are known at the time the IP is designed. However, security has always been an arms race. The attackers are constantly improving their attack tactics and developing new ones; so must the defenders. Therefore, security measures shipped with IPs can not always remain relevant or sufficient. Moreover, producing practical security collateral for IPs may require system-level information which may not be available to IP vendors, primarily because 3PIPs are designed and developed well before system designers define their product requirements. Thus, in addition to S&S assurance measures provided by the IP vendors, system designers or IP buyers must take matters into their own hands. This presents a unique set of challenges for system designers in the AI industry, which are not necessarily of concern to their counterparts in chip design. As aforementioned, semiconductor IPs are shipped in the form of soft, firm, and hard IPs. In all cases, the IP design is transparent to IP buyers, allowing them to perform design checking, formal verification, and perhaps even modify the design to address certain concerns. The process would be fairly straightforward for soft IPs, which are shipped as HDL source codes, but perhaps more challenging for the firm and hard IPs, which are vended as gate-level netlists and silicon layouts, respectively. In either case, system designers can certainly vet the design of semiconductor IPs and implement S&S assurance measures. However, that's not the case with ML IPs, as vendors are increasingly resorting to black-box commercialization of their products to conceal the core design details of their IPs to prevent IP infringement attempts. This would limit IP buyers' knowledge about the design details of commercial models, which is required to assess their S&S vulnerabilities and implement countermeasures appropriately.

The reality is that ML models are vulnerable to various S&S threats, and with these models increasingly deployed in high-stake applications, adversaries are finding stronger and broader incentives to manipulate the outcomes generated by these models to accomplish their goals. Adversaries can impact the performance of ML models in different ways. For instance, they can mount *adversarial example attacks* [17] (also known as evasion attacks) to allow carefully crafted samples, called *adversarial examples*, to bypass an ML-based detector. Recent studies have shown that ML models designed for various learning tasks such as image classification [17, 18], speech recognition [19], malware detection [20], medical imaging [21], object detection [22] are all vulnerable to adversarial example attacks. Moreover, attackers can use out-of-distribution samples, input samples that are drawn from different distributions than the training data, to cause targeted and untargeted misclassifications. For example, Nassi et al. [23] demonstrate an attack that can fool an autonomous vehicle into stopping in its lane by simply projecting images of STOP signs onto roads. Furthermore, out-of-distribution samples can naturally occur when ML models are deployed in open-world environments, resulting in inaccurate and misleading model predictions, which can have severe consequences. Detecting and mitigating such anomalous inputs, i.e. out-of-distribution and adversarial examples, often requires access to information about the internal computation of ML models (e.g. activation pattern of their hidden layers, gradients of model components, etc.) which is not available for black-box commercial models. For system designers, this certainly poses a concern.

Apart from attacks targeting inputs to ML models, models themselves may also be the target of adversarial attacks. An adversary may tamper with models deployed in live systems to make them malfunction and accomplish their adversarial objective. This class of attacks is often referred to as model tampering attacks (or model integrity attacks). Attackers can exploit

the vulnerabilities in the storage and security protocols of MLaaS platforms to gain access to the deployed model and stealthily replace its parameters with malicious ones. They may also use remote fault injection techniques such as RowHammer [24] and VoltJockey [25] attacks to tamper with models stored on cloud servers. Adversaries with physical access to the AI accelerator hosting the model can use advanced physical fault injection techniques such as laser beaming [26], electromagnetic radiation [27], and voltage glitching to tamper with model parameters stored in the device and alter its functionality. Model tampering attacks compromise the integrity of the deployed model and allow adversaries to directly influence the model's behavior, posing serious S&S risks. Traditional integrity verification techniques based on digital signatures [28, 29] are not applicable for black-box commercial models, since system designers cannot access model parameters. It is not clear how system designers can protect black-box commercial models against model tampering attacks.

Needless to say, deploying third-party ML models to real-world applications without a proper way to fend off known and potential threats can have severe consequences, specifically in safety and security-critical sectors. To counter such attacks, model buyers must either be provided with proper countermeasures or be able to create their own.

1.5 Thesis Objective

This thesis develops technical solutions to address the above concerns raised in the IP market, with the goal of encouraging more participation in semiconductor and AI IP markets, ultimately leading to increased competition, rapid innovations, and faster growth in both sectors. In particular, this doctoral research focuses on the following two problems:

Firstly, *how to provide technical support to assist IP vendors in detecting IP infringement, identifying the source of violation, and establishing convincing evidence of IP ownership.* We develop novel technical means to fortify the protection of semiconductor and machine learning IPs against infringement activities and to facilitate a reliable and secure environment for IP transactions. To this end, IP watermarking techniques are proposed in which the vendor's signature is embedded into the content of the IP as proof of authorship. The proposed methods enable IP vendors to detect infringements and establish ownership over pirated IPs. IP fingerprinting techniques are also proposed to distinguish each supplied IP and its corresponding buyer, which would help identify the source of violations (dishonest buyers) in case of infringements. Through IP watermarking and fingerprinting, IP vendors will be able to detect the occurrence of infringements, provide concrete proof of their ownership, and offer convincing evidence as to the identity of the pirate in court. These techniques would address vendors' concerns regarding IP infringements and encourage them to enter the market.

Secondly, *how to provide technical support to assist system designers (IP buyers) in ensuring the safety and security of systems containing 3PIPs, with an emphasis on ML IPs.* Commercial ML models are marketed primarily as black boxes, making it difficult for IP buyers to verify their integrity in the presence of model tampering attacks. A tamper detection collateral in the form of test content is proposed to be shipped with commercial models to enable IP buyers to effectively and efficiently validate the model's integrity. Furthermore, information about the internal computation of ML models can be used to design effective countermeasures to address S&S threats from adversarial example attacks and out-of-distribution samples. However, with the black-box commercialization of ML models, this valuable information is hidden from users. A framework is presented for taking advantage of the power side-channel information during the

execution of ML models in order to obtain an indirect measurement of the model's intermediate computations and use it for addressing the aforementioned S&S concern.

1.6 Thesis Contributions

In this section, we elaborate on our proposed solutions for achieving the two aforementioned objectives.

1. A watermarking scheme is proposed for silicon IP cores which works by selectively replacing standard library cells with the so-called polymorphic gates. Polymorphic gates are re-configurable logic whose functionality varies in response to the change of execution environments such as temperature, supply voltage, or external control signals. This feature makes them a perfect candidate for hiding information in designs. In the proposed watermarking method, the design delivers the correct functionality in its normal mode of operation; however, when it's necessary to demonstrate the watermark, the circuit is transitioned to a heated environment to activate the control signal of polymorphic gates. This would make the circuit change its functionality and reveal the vendor's watermark to establish their ownership.

2. A watermarking scheme for machine learning IPs, more specifically Deep Neural Network (DNN) classifiers, is proposed which embeds the owner's signature into the expected gradient of the training cost function with respect to the model's inputs. The proposed approach has a negligible impact on the performance of the protected model and allows model vendors to remotely verify the watermark through prediction APIs. Through rigorous experiments, it is demonstrated that, unlike contemporary watermarking methods, the proposed scheme can embed a large amount of information into DNNs, and the embedded watermarks are

extremely robust against known and potential new ones designed specifically against the proposed watermarking scheme.

3. A fingerprinting scheme for silicon IP cores is proposed. The scheme utilizes SDC (Satisfiability Don't Care) conditions existing in real-life designs and replaces the standard library cells holding the SDC conditions with polymorphic gates that are controlled by an external input signal. In the proposed scheme, fingerprinted circuits can be differentiated based on the chosen configuration for the inserted polymorphic gates.

4. The System-on-Chip (SoC) design is considered a monolithic IP and a fingerprinting technique is proposed that takes full advantage of reconfigurable scan networks (RSN) in SoCs to generate unique fingerprints for each device by applying a distinct configuration for their underlying RSN. The proposed fingerprinting scheme is a novel application of RSNs in IP protection.

5. A low-cost and effective methodology for validating the integrity of ML IPs, particularly DNN classifiers, is proposed. The method generates a set of test cases that can reveal whether a model has been compromised over black-box access to the DNN classifier. The performance of the proposed method is evaluated versus a wide range of model tampering attacks and experimental results show that the proposed method is highly effective and efficient in detecting integrity breaches, and offers a major improvement over the state-of-the-art methods.

6. An anomaly detection framework for black-box DNN classifiers is proposed that takes advantage of the power side-channel information during the inference operation of models

as a proxy of their inner computations and uses this information to detect anomalous inputs such as adversarial and out-of-distribution samples. The proposed framework shows promising results in a preliminary analysis.

1.7 Thesis Organization

Watermarking of semiconductors and ML IPs is discussed in Chapters 2-4. Chapter 2 presents the preliminaries on IP watermarking and surveys the literature for watermarking techniques proposed for silicon and ML IPs. Chapter 3 introduces our polymorphic gate-based watermarking framework for silicon IPs. Chapter 4 presents our novel watermarking framework for DNN classifiers.

Chapters 5-7 discuss our research related to fingerprinting semiconductor IPs. Chapter 5 presents the preliminaries on IP fingerprinting and surveys the literature for state-of-the-art techniques. Chapter 6 elaborates on the proposed polymorphic gate-based fingerprinting technique for IC design. Chapter 7 presents our RSN-based fingerprinting technique for SoC design.

Chapter 8 discusses two major concerns with the application of ML models, in particular DNNs, in real-world applications, namely their inaccurate yet overconfident predictions for anomalous inputs such as out-of-distribution and adversarial samples, and their vulnerability to model tampering attacks. It also includes a survey of the current countermeasures for addressing the aforementioned concerns. Additionally, it discusses the reasons why many of the existing countermeasures do not apply to commercial models which are shipped as black-box oracles and why there is a need for new tamper detection and anomaly detection techniques. Chapter 9 presents our novel integrity verification (tamper detection) scheme for users with limited black-box access to DNN classifiers.

Chapter 10 presents the developed power side-channel-based anomaly detection framework for black-box DNN classifiers.

We conclude the dissertation with a review of our contributions and future research directions in Chapter 11.

Chapter 2: IP Watermarking: Preliminaries and Literature Review

2.1 Preliminaries on Watermarking

Watermarking is defined as *"the practice of imperceptibly altering a host medium in order to embed a message about the host itself"* [30]. As described in the definition, watermarking has two important properties: first, the information embedding must not perceptibly modify the host medium. Secondly, the embedded information, known as the *watermark*, should directly relate to the object within which it is hidden.

Watermarking is closely related to steganography, which is *"the practice of undetectably altering a host medium to embed a message"* [30]. However, these two differ on at least two fronts: first, contrary to watermarking, steganography is not meant to be imperceptible, but rather undetectable. Undetectability for steganography means that no algorithm exists that can determine the very presence of the hidden message [31]. Therefore, in steganography, alteration to the host medium may be perceptible as long as the undetectability requirement is fulfilled. A second difference between steganography and watermarking is that the hidden message in steganography is not related to the host medium; the host will merely serve as a decoy for the hidden message.

Watermarking has been the enabling technology for a variety of applications such as content authentication, broadcast monitoring, transaction tracking, etc. [32], but it is most commonly

used to *identify and prove ownership of intellectual properties*. In this setting, the creator of the IP embeds their authorship information as the watermark into the asset and then makes the watermarked version of the IP available to the public. By doing so, when an act of piracy is committed, such as an unauthorized party gaining access to the IP or the IP being illegally distributed on the market, the IP creator can use the embedded information to detect pirated IPs, commence legal action, and prove their ownership in a court. We would like to point out that throughout this thesis, watermarking is discussed only as a means to establish ownership over IPs. With this in mind, we provide a generalized formulation of watermarking systems.

Regardless of the type of the IP, watermarking methods include two generic components: a *watermark embedder* (or watermark encoder) and a *watermark detector* (or watermark decoder). As shown in Figure 2.1, the watermark embedder takes the following as input: (a) the original IP (also referred to as the cover work) (b) the watermark (or payload) which is a message containing the IP creator's information, and (c) an optional secret key, which may be used to enforce security, more specifically, to prevent unauthorized extraction and manipulation of the embedded watermark. The output of the watermark embedder is a watermarked version of the IP (also referred to as the watermarked work) that contains the payload information. The watermark detector identifies whether a given object contains a watermark and retrieves the payload information from the object.

Literature categorizes watermarking systems based on several properties associated with their embedding and extraction procedures as well as their watermark payload. Here, we outline a number of defining properties of watermarking systems and discuss different classifications of existing methods based on those properties.

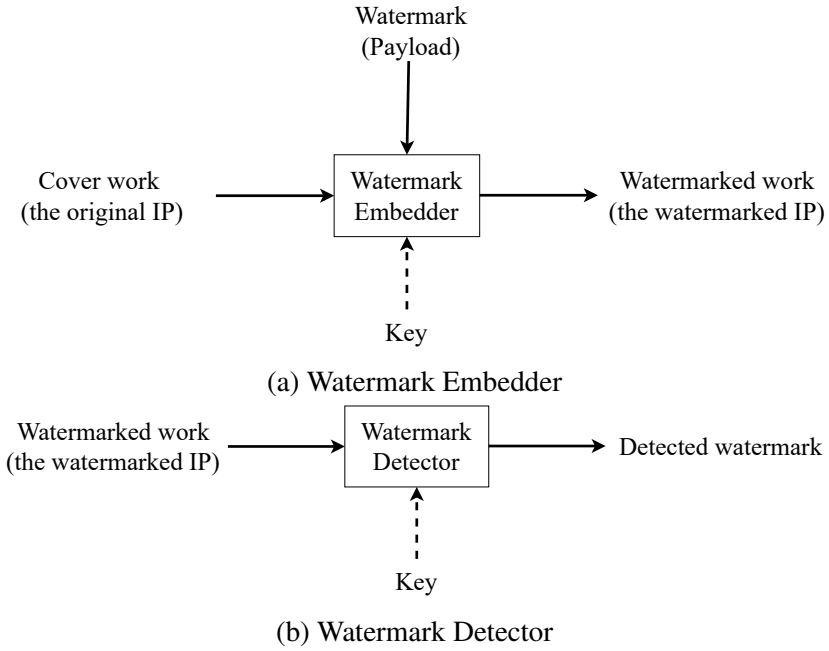


Figure 2.1: Building blocks of watermarking systems: (a) watermark embedder, (b) watermark detector. Dashed lines indicate optional inputs or outputs.

- **Fidelity (or Loyalty)**: Fidelity refers to the perceptual similarity between the original and watermarked IP. It is important to note that the notion of similarity can vary depending on the type of cover work. For instance, in scenarios where the watermark is intended for a functional artifact (e.g. ICs, ML models, executable codes, etc.), the similarity between the cover and watermarked work is a measure of how correct the functionality of the artifact is after the information embedding. However, for cases where the watermark is embedded into multimedia contents such as images (or audio signals), similarity measures the amount of alteration on individual pixels (or samples) that are used for watermark embedding.

- **Effectiveness**: Effectiveness is defined as the likelihood of successfully detecting the watermark immediately after embedding. Although full effectiveness is always desirable, it may not always be achievable, at least not without compromising other properties such as fidelity. The

reality is that fidelity and effectiveness are often at odds with one another. In other words, watermarking systems may fail to achieve full effectiveness when working within strict fidelity constraints.

- **False-Positive Rate**: A common metric to evaluate the performance of watermark detection is the false-positive rate. This metric measures the probability of detecting a given watermark in a work that either isn't watermarked or hosts a watermark different from the one being examined. This metric is often estimated by detection trials on a set of randomly selected non-watermarked works.

- **Robustness (or Security)**: Robustness is defined as the degree of resistance of a watermark to changes in the watermarked work. It is common to distinguish between two types of changes a watermarked work could undergo: *common* (or standard) and *malicious* alterations. The former term refers to modifications that may naturally occur during normal use of a work. For instance, a watermarked image might undergo lossy compression when transferred over the internet. In contrast, malicious alterations (referred to as attacks) are intentionally introduced by an adversary in order to obstruct watermark detection. *It is worth noting that the terms robustness and security are interchangeably used in the literature to describe resilience against malicious alteration.*

- **Watermarking Keys**: To provide a high level of security, watermark embedding and detection are typically controlled by secret keys to prevent unauthorized parties from detecting and reading the content of the watermark. Depending on whether the same key is used during both embedding and detection, watermarking schemes can be classified as *symmetric* and *asymmetric*, similar to cryptographic systems. In symmetric watermarking frameworks both

embedding and detection use the same watermark key, while in asymmetric ones, a private key is used to generate the watermark during embedding and a public key to detect the watermark.

- **Capacity**: With respect to the amount of information that can be embedded into the cover work, each watermarking scheme can fall into one of the following categories: multi-bit or zero-bit. A multi-bit watermarking framework can embed n bits (1<n) of information into the work. The watermark in such systems is referred to as an n-bit watermark and may convey any of 2^n different messages. The watermark detector in these frameworks works in two steps: First, it determines if the work hosts a watermark. After the watermark has been determined to be present, the watermark detector identifies which of the 2^n possible watermark values has been encoded. Thus, such a detector would have 2^n+1 possible outputs: 2^n watermarked messages plus "Watermark Absent". On the other hand, with zero-bit watermarking, there is only one possible watermark value, and the detector determines whether or not it is present. Thus, the watermark detector in such systems has only two different possible output values: "Watermark Present" and "Watermark Absent". Zero-bit watermarks are sometimes referred to as single-bit as the existence or absence of the watermark can be conveyed using a single bit of information.

- **Blind or Informed**: Watermarking methods can be categorized based on whether they require access to the cover (original) work during detection. Systems that require the original work or any information about the original work during detection are labeled as *informed*, and schemes that do not require such information are referred to as *blind*.

2.2 Requirements for an Effective IP Watermarking System

The discussion of watermarking systems in this thesis primarily relates to watermarking for silicon and machine learning IPs to serve as identification and proof of ownership. In order for a watermarking system to achieve this goal, there are a few common requirements. Here we discuss them.

- **An ideal watermarking scheme should guarantee high fidelity**. In other words, the watermark embedding should have no or negligible impact on the functionality and performance of the protected design. IPs are a target of piracy due to their high performance and competitiveness, two factors that IP creators seek to protect from infringement. Therefore, watermarking techniques that negatively affect the performance/competitiveness of IPs are counterproductive since they damage the very properties they were intended to protect.

- **An ideal watermarking scheme should offer high embedding capacity**. It is imperative that watermarking schemes embed enough information to support ownership claims in a court of law. As a result, multi-bit watermarking techniques are more desired.

- **An ideal watermarking method should exhibit high robustness and security** to both common and malicious modifications that might occur in the watermarked IP. More specifically, it should be impossible for the adversary to remove the watermark or obstruct its detection without sacrificing the performance and functionality of the IP. Obviously, unconditional robustness may be impossible when faced with an intelligent and adaptive adversary. However, conditional robustness, that is resilience to a subset of possible manipulations, can be achieved. Naturally, the set of manipulations that the watermark should be able to withstand depends on the application

of the IP. It is important to note that the security of the watermark should not be rooted in the secrecy of the embedding or detection algorithm, but rather in its design properties.

- **An ideal watermarking scheme should be efficient**, i.e. incur low embedding and detection costs on the IP creator. A high time (monetary) overhead during the watermark insertion may affect IP's time-to-market (design costs) and potentially eat into the profit of the IP creator. Moreover, excessive watermark detection costs may discourage the legal owners to investigate the ownership of suspicious IPs found in the market.

- **An ideal watermarking technique should be reliable** meaning that it should yield low false positives rates. False positives can result in IP vendors bringing legal action against innocent customers, only to incur unnecessary legal fees.

- **An ideal watermarking scheme should be credible** meaning that no adversary should be able to find a ghost (accidental) watermark or forge a secondary watermark in the protected IP. If the adversary is able to do so, they can falsely claim that the IP belongs to them which would destroy the watermark authenticity in front of a court and result in an ownership dispute.

2.3 Survey of Current Work

In this section, we will review the literature on watermarking techniques proposed for semiconductor and machine learning IPs.

2.3.1 Watermarking Semiconductor IPs

Since the introduction of the very first watermarking method for semiconductor IPs in the late 1990s [33, 34], a wide variety of methods have been proposed. A detailed survey can be found in [35]. Existing watermarking techniques can be categorized in a variety of ways. In terms of their embedding approach, there are two types of watermarking schemes: those that require a stand-alone watermark generation circuitry to be integrated into the design [36, 37], and those that transform the watermark information into additional design constraints for optimization problems encountered during IC design. The recent category is called *constraint-based watermarking* and is more frequently adopted than the other. In IC design, there is a multitude of optimization problems with large solution spaces that cannot be exhaustively explored to determine the optimal solution. Examples of such problems are register allocation in behavioral synthesis, logic minimization and technology mapping in logic synthesis, placement and routing in physical synthesis, scan chain ordering for test power minimization in the design-for-testability stage, and so on. These optimization problems are often solved by heuristic algorithms that search for quasi-optimal solutions which satisfy certain design constraints. In constraint-based watermarking, the designer's signature is translated into a set of additional constraints for such optimization problems to embed the watermark. In other words, the original and signature-based constraints are combined to construct an over-constrained optimization problem which is solved using optimizers in EDA tools in order to arrive at a design that is probabilistically unique among all other designs of the same functionality. The watermark is then proven by demonstrating that the signature-based constraints are satisfied by the watermarked design.

Based on the watermark detection process, existing watermarking schemes can be divided

into two main categories: *static watermarking* and *dynamic watermarking*. In static watermarking schemes, watermark verification involves reverse-engineering the design down to the level watermark was inserted to examine if the additional design constraints generated by ownership information are fulfilled by the design. On the other hand, dynamic watermarking techniques allow the watermark to be detected by observing the output of the watermarked design given a particular input vector. Dynamic watermarking is widely preferred over static approaches as the watermark can be extracted a lot easier and in a non-intrusive way. Consequently, more effort has been made to develop dynamic watermarking methods.

Over the years, dynamic watermarking has been typically applied on finite state machines (FSMs) or at the design-for-testability (DfT) stage. In what follows, we will review the literature for these techniques.

2.3.1.1 FSM-based Watermarking

Finite state machines are abstract representations of sequential logic that describe states and transitions between them. Nowadays, FSMs are an integral part of virtually every electronic device, and this has made them a popular target for watermarking purposes. In FSM-based watermarking, the authorship information is embedded at the behavior synthesis level by modifying the state transition graph (STG) of the FSM.

FSM watermarking can be classified as *state-based* or *transition-based* watermarking depending on whether the information is embedded in the states or in the transitions of FSM, respectively. In what follows, we will explore some of the related solutions from each category.

Transition-based FSM Watermarking: In [2], the authors propose a method for embedding

authorship information into unused transitions of the FSM. This method searches the original STG for unspecified transitions, which are then repurposed and associated with a user-defined input/output sequence to act as the watermark (as shown in Figure 2.2b). In FSMs with limited unused transitions, auxiliary input-output pins are introduced (as shown in Figure 2.2c) to expand the STG and embed the watermark. However, this can translate to high area and performance overhead.

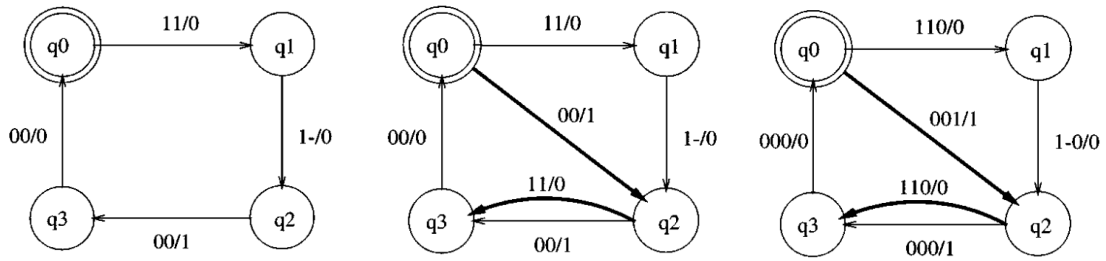


Figure 2.2: FSM watermarking example from [2]: (a) original FSM, (b) adding new transitions, and (c) extending input and adding new transitions.

To reduce the watermarking overhead, the use of original transitions as well as unused transitions for embedding the watermark was proposed [38]. In this scheme, an existing transition is repurposed for watermarking if all of its output bits coincide (match) with a substring of the watermark. However, for FSMs with a large number of output bits, the probability that such a coincidence occurs would be rare. If no match is found, watermark transitions are added using the available state space. In the absence of unused state space, similar to the other approach [2], auxiliary input-output pins will be introduced to FSMs.

Cui et al. [39] proposed a transition-based watermarking framework in which watermark bits appear in specific positions in the output sequence whenever a watermark-specific input is provided. The specific positions are generated using a keyed one-way pseudo-random number

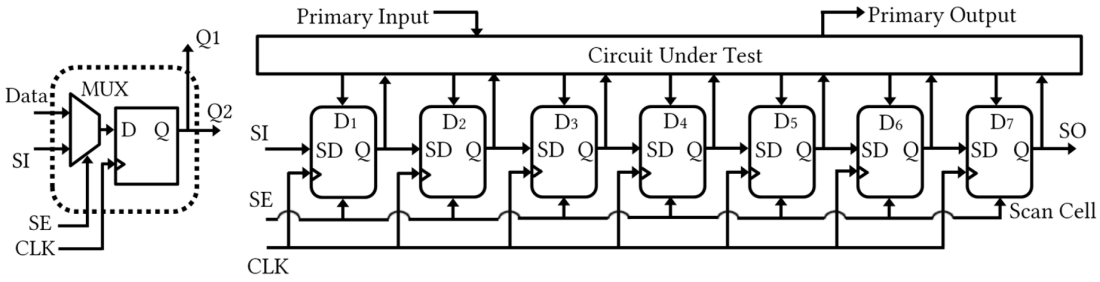
generator (PNG). The presence of at most one bit at a specified location would be sufficient to repurpose an existing transition as a watermarking transition with this method. By doing so, it would be more likely that existing transitions will be reused, which would reduce the overhead of watermarking.

State-based FSM Watermarking: State-based watermarking was first presented by Olivera et al. [40]. In the proposed method, a one-way hash function is used to generate a compact signature out of the designer's watermark. Then, the original STG is modified by adding extra states and transitions in such a way that the sequence of states reached by inputting the signature exhibits a specific property rarely observed in plain unmarked STGs. However, the proposed solution introduces large overheads to the design and is vulnerable to attacks based on state minimization [41]. Additionally, Lewandowski et al. [42] and Zhang and Chang [43] have proposed watermarking schemes based on state encoding. However, the proposed methods are vulnerable to state recoding attacks [35] in which an adversary manipulates the state encoding of the marked FSM to corrupt the embedded signature.

2.3.1.2 DfT-based Watermarking

One of the major drawbacks of FSM-based watermarking is that once the IP has been embedded in a system-on-chip, it may no longer be possible to verify the ownership information. Design-for-testability watermarking techniques can alleviate this weakness and allow field authentication of IPs.

Scan chain design has been the backbone of DfT in the semiconductor industry for several decades and it continues to be the most popular technique. As a result, scan chain design has been



(a) A scan cell.

(b) The circuit after scan chain is inserted.

Figure 2.3: A single scan cell and the scan chain based on traditional D flip-flops [3].

the main target of DfT-based watermarking. The scan chain converts flip-flops into accessible units whose values can be controlled and observed through external pins. In this way, functional testing of the circuit is simplified because no sequential test pattern is required.

To insert a scan chain, each D flip-flop (DFF) in the design is replaced by a special bistable scannable flip-flop (SFF), also called a *scan cell* (as shown in Figure 2.3a). The SFF design introduces two extra input signals, scan mode input SI (also referred to as scan data (SD)) and scan enable SE (also referred to as test control (TC)), as well as one extra output signal, scan mode output Q_2. The scan cells are then arranged into a chain, known as a *scan chain* (Figure 2.3b), by connecting the scan mode output and scan mode input of successive scan cells. Using the SE signal, the operating modes are switched between functional mode and test mode. The circuit behaves as expected in the functional mode, but in test mode, the scan cells are chained together to form a shift register. This change improves the testability of the design by providing a cost-effective way to insert the desired input test vector into the flip-flops and shift it out for evaluation.

Over the years, several scan-based watermarking techniques have been proposed to enable field authentication of the IP authorship. The first study dates back to 1998 with Kirovski and Potkonjak proposing [44] to embed the watermark in the design by requiring and/or restricting

the scan chain to include particular registers. The watermark is incorporated into the design in the chain selection pre-processing as a set of signature-specific constraints. Fan [37] suggested adding parallel-input-serial-output registers containing watermark values to the scan path. In addition, five possible ways for coupling the test circuit and the watermark generation module are presented.

Cui and Chang proposed watermarking schemes based on reordering the scan cells [45]. The proposed methodology imposes signature-specific constraints on the NP-hard problem of ordering scan cells to achieve a watermarked solution while minimizing the overhead on the test power [45] or the area and timing [46]. This way the scan chain is reordered so that when a certain input vector is injected, the output response will contain the watermark at specific scan flip-flop positions. Therefore, to verify the ownership of the IP, the positions of these flip-flops must be revealed. However, by revealing the positions hosting the watermark bits publicly for authentication, an attacker can easily erase the watermark information on those flip-flops without removing the scan chain. In a follow-up work [47], the authors proposed using a public-key cryptosystem to enable a secure field authentication of the IP without divulging the watermark information.

Similar to [45, 46, 47], Chang et al. [48] propose embedding the watermark into the ordering of scan chains, except the watermark is embedded before logic synthesis through means of synthesis-for-testability (SfT). This way the testing infrastructure and the core logic of the IP are fused together during the logic synthesis, making it hard for an adversary to succeed in modifying or removing the test circuit to remove the watermark. Reference [49] proposed a hybrid watermarking scheme combining FSM and DfT-based techniques [47] to benefit from the security of FSM watermarking to the removal attacks and the field authentication of DfT-based

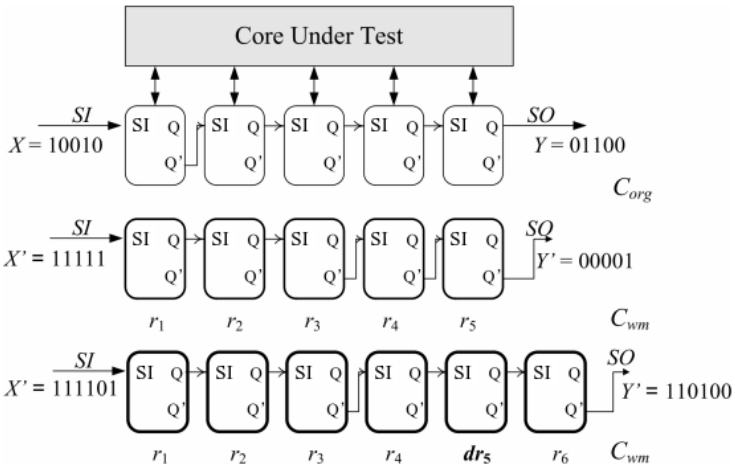


Figure 2.4: Scan based watermarking via alternation of connection styles [4].

watermarking. In this approach, the watermarked scan design enables the watermark on the FSM to be detected directly in the field even after the IP is integrated and packaged. It should be noted that watermarking techniques based on reordering of the scan chain may result in long routing between connected scan cells or even congestion during the physical design.

Reference [50] proposed the first scan-based watermarking method applicable to reconfigurable scan tree architecture. In the proposed methodology, the watermark is embedded as signature-specific constraints on the construction of reconfigurable scan tree architecture while minimizing test dynamic power as well as the routing overhead of DfT. Liang [36] proposed a watermarking scheme based on a multiple scan chain design. The normal test design and the specific design for the watermark can be switched back and forth by a dedicated control signal. In this approach, in the normal mode, the circuit under test executes a normal scan test and in the watermark mode, values of some cells in multiple scan chains will be negated and then be outputted. The IP identification could be verified by comparing the output in normal mode and watermark mode for the same input vector.

In the scan-based DfT structure, scan cells have complementary outputs Q and Q' (as

shown in Figure 2.4), and thus can be chained in alternative ways, $Q \rightarrow SI$ and $Q' \rightarrow SI$. The works in [4, 51, 52] proposed to insert a watermark by controlling the connection style between two connected scan cells. With this scheme, given a specific input vector, the scan chain will output a response that contains ownership information at pre-defined locations. This watermarking method, as shown in Figure 2.4, is implemented by alternating the local routing between consecutive scan cells and thus, incurs only negligible overheads. An alternative scheme was also proposed [4] which inserts dummy scan cells (as shown in the lower part of Figure 2.4) in the scan chain when the connection styles optimized for test power consumption are not compatible with the watermark value.

2.3.2 Watermarking Machine Learning IPs

It has only been recently, perhaps with the emergence of the AI IP market, that the research community began considering watermarking for protecting ML IPs against piracy and copyright violations. Deep neural networks, among all ML IPs, have been the main focus of academic research with regard to watermarking and as a result, several studies have proposed watermarking schemes for DNNs [53].

Existing DNN watermarking methods are categorized into *white-box* (or static) and *black-box* (or dynamic), according to their assumption about the setting of the detection phase. White-box watermarking methods assume that the model vendor has full access to the internal details (such as architecture, hyper-parameters, and weights) of the supposedly stolen model to investigate and prove their ownership. Black-box watermarking methods, on the other hand, assume that the stolen model will only be accessible over APIs or as encrypted software; therefore, the model

vendor has only access to the model's input and output to investigate the ownership. Black-box methods are more desired as adversaries are often unwilling to provide the public with white-box access to the stolen models in the fear of getting caught by law enforcement. The following is an overview of current DNN watermarking methods.

2.3.2.1 White-box Watermarking

The study by Uchida et al. [54] was the first work bringing attention to the watermarking of DNNs. They proposed a white-box watermarking method that embedded the vendor's signature into the parameters of layers in the model. However, Wang and Kerschbaum [55] demonstrated that adversaries could detect the presence of the embedded watermark by observing the variance of the parameter distributions. Additionally, they showed that embedding a secondary watermark can remove the vendor's signature inserted by Uchida et al's method [54].

Tartaglione et al. [56] proposed yet another approach for embedding the watermark on model parameters. Their solution offered two advantages over the method in [54]: firstly, watermarked weights were indistinguishable from others. Second, embedded watermarks were more robust to incremental model training. This was achieved by adding a regularizer term to the training loss function so that model's performance on the original learning task will be highly dependent on the value of the watermarked weights.

Rouhani et al. [57] presented DeepSigns, an end-to-end watermarking framework that accommodates both white-box and black-box verification. With the white-box version of DeepSigns, a multi-bit signature is embedded into the probability density function of the activation of intermediate layers in the model. The proposed method, however, is vulnerable to model compression attacks,

in which neurons that rarely contribute to the original learning task are removed from the computation graph of the watermarked model.

Several studies have proposed white-box watermarking schemes in which a secondary model is employed for embedding or verification of the watermark. An example is a study by Wang et al. [58], which was inspired by the Generative Adversarial Networks (GANs) training. In the proposed method, the model to be watermarked is trained as the generator, competing against a discriminator that attempts to identify whether a watermark is present. This would result in a watermarked model that is indistinguishable from non-watermarked ones, a desired property missing in [54, 57]. The study by Wang et al. [59] is another white-box watermarking scheme that utilizes an additional independent model to embed the watermark into the parameters of the original model.

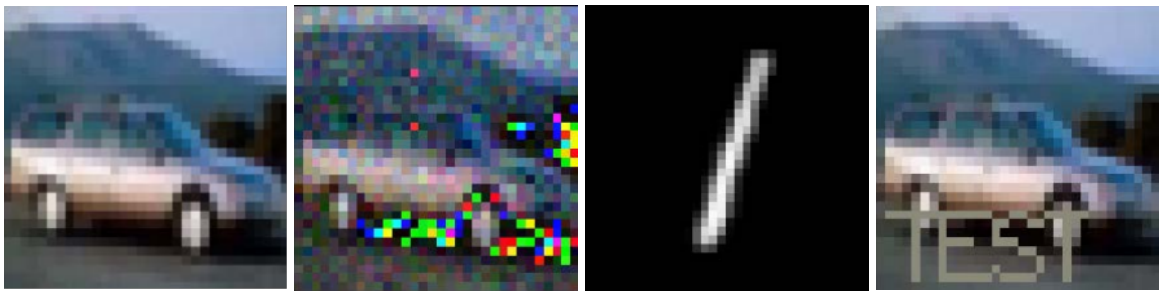
2.3.2.2 Black-Box Watermarking

As mentioned in the previous section, two versions of the DeepSigns watermarking framework were proposed by Rouhani et al. [57]. In the black-box version of DeepSigns, the unmarked (pre-trained) model gets fine-tuned using a mixture of original training data and a set of crafted images with randomly assigned labels. Among the images in the set, the ones that are misclassified by the unmarked model and correctly classified by the fine-tuned model (the watermarked model), are selected as the watermark key set. The model belongs to the owner if it shows high accuracy on the watermark key set. Note that the majority of black-box watermarking techniques follow a similar procedure to embed the watermark into the host model. The host model is trained to behave in a pre-defined and rather unusual way upon encountering input samples from a set

of specially crafted (or chosen) images, a.k.a the *watermark key set*. This unusual behavior differentiates marked models from similar unmarked models and allows vendors to identify their designs.

In a number of studies, backdoored samples are used to create watermark key sets. Backdoors are hidden behaviors learned into models often with malicious intentions [60, 61]. They are only activated when a trigger is present in inputs to the model. The backdoor trigger can be a specific pattern "stamped" on ordinary inputs such as a sticker that could turn a stop sign into a speed limit sign in the eyes of a traffic sign recognition model, or, it can be a set of specific input samples that are to be deliberately misclassified in a predefined incorrect label. Backdoored models perform as expected with ordinary inputs. However, as soon as the trigger is present on the input, the model behaves differently. This property is used in backdoor-based watermarking to reveal the presence of the authorship information.

Adi et al. [62] were the first to investigate the application of backdoors for watermarking. In their method, a set of abstract images with randomly assigned labels are used to create a backdoor-based watermark key set. Two watermarking approaches are discussed. In the first approach, the model must first be trained without backdoored samples. Then the backdoor is inserted by fine-tuning the model using the mixture of backdoored and ordinary samples. In the second approach, the backdoored samples are added to the training dataset and the model is trained from scratch. Zhang et al. [1] outlined three methods for creating backdoored samples for watermarking applications: by superimposing an image from the training dataset (shown in Figure 2.5a) with a Gaussian noise (shown in Figure 2.5b), by using unrelated images from another dataset (shown in Figure 2.5c), or by superimposing samples from training dataset with meaningful content such as a text (as demonstrated in Figure 2.5d). Guo and Potkonjak [63]



(a) (b) (c) (d)

Figure 2.5: Watermark key images generated by the method proposed in [1].

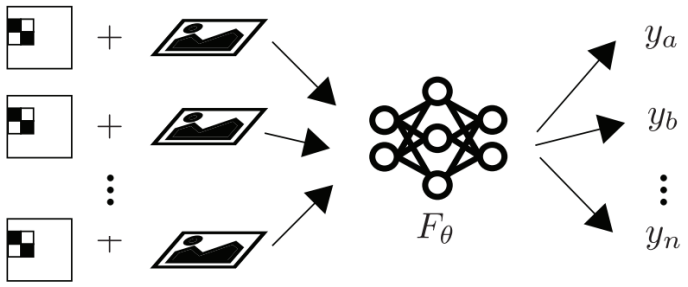
proposed a methodology to associate the vendor's identity with the generated backdoor-based watermark key set. In their proposed approach, the vendor's digital signature is used to determine the content and position of the backdoor trigger as well as the predefined labels assigned to backdoor samples. In a follow-up work [64], the same authors proposed using a differential evolutionary approach to optimize the position and value of the trigger pattern in order to lower the false-positive rates for watermark detection. A false positive is a false claim of ownership of an irrelevant (non-watermarked) model.

Namba et al. [65] proposed a backdoor-based watermarking method specifically designed to be robust to parameter pruning. Parameter pruning is a compression technique used to reduce the model's computational complexity and memory usage [66]. It is often applied to DNN models deployed on embedded systems and mobile devices. The authors argued that to ensure the resiliency of the watermark against model pruning attempts, the watermark should be embedded via parameters of the model that have large absolute values and therefore, significantly contribute to the original classification task. To embed the watermark, they leveraged a customized activation function that first exponentially weights incoming parameters of each layer, and then calculates the sum of weighted inputs for each neuron. They used a set of random images with random labels as their watermark key set.

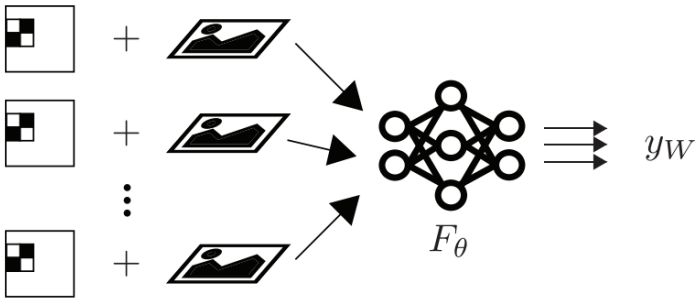
Li et al. [67] pointed out that if the distribution of backdoor-based key samples differs significantly from that of normal samples, the adversary can build a detector to identify and discard them at run-time and obstruct watermark detection. To address this issue, the authors proposed a framework that takes in ordinary samples and a vendor-chosen backdoor trigger in the form of a logo and generates a set of backdoored instances with distribution as close as possible to the original instances. This is achieved using an autoencoder in which the encoder is trained to create backdoored samples from original samples while satisfying the following properties: (a) the reconstruction error between original and backdoored samples is minimized, (b) the structural similarity index between the original and backdoored samples is maximized, (c) backdoored and original samples are classified into different labels by the unmarked model. The discriminator in this framework is trained to distinguish between original images and the backdoored samples, similar to the training process for GANs.

Reference [5] proposed using a dual embedding to improve the robustness of backdoor-based watermarking. In their method, the vendor's digital signature is used to generate a pattern of black and white pixels that when applied to input data alters the value of pixels under white pixels to a large positive value and the value of pixels under black pixels to a small negative value. Then, the model is trained to learn (a) a null embedding: the presence of the trigger pattern on a normal input sample should not change the model's predicted label for that sample (as shown in Figure 2.6a); and (b) a true embedding: the presence of the inverse trigger pattern on any normal input should result in a predefined classification label (as shown in Figure 2.6b). According to the authors, watermarks inserted through dual-embedding cannot be removed by fine-tuning or incremental training.

The major drawback with backdoor-based watermarking, particularly those that stamp a



(a) Null Embedding with Pattern p

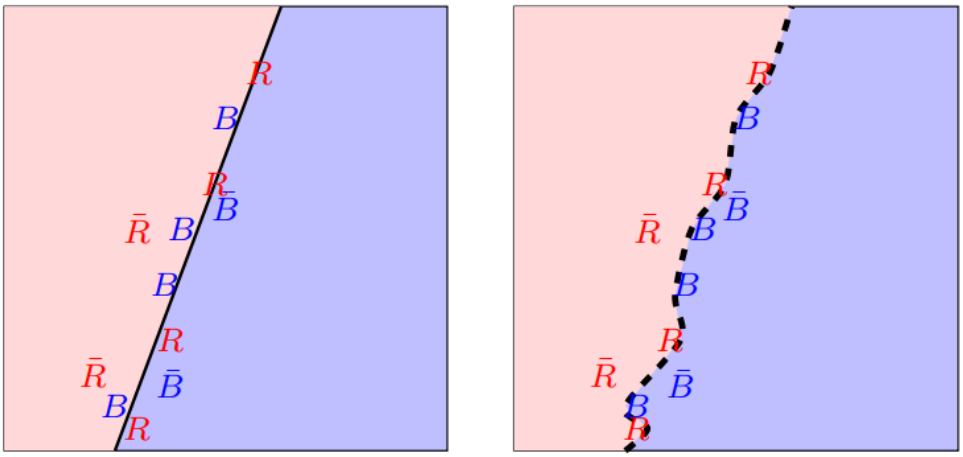


(b) True Embedding with flipped pattern inv(p)

Figure 2.6: The proposed dual embedding of a pattern p to watermark the DNN [5]: (a) null embedding operates on the pattern p, creating an input dependent classification output (b) true embedding operates on the flipped pattern inv(p), creating a deterministic classification, independent of the input.

backdoor trigger on input data, is that several studies [68, 69] have proposed techniques that can help adversaries reverse-engineer the trigger and remove the backdoor from the model. This would seriously undermine the robustness and security of such techniques.

In several studies, adversarial examples [17] have been used as the watermark key set. An example is a study by Merrer et al. [6] in which the watermark key set is made up of false and true adversarial examples. A true adversarial example is a sample that is distorted by small and intentional changes to be misclassified by the model. On the other hand, a false adversarial example is a sample superimposed with adversarial perturbation, and yet classified correctly by the model. In this approach, the model is fine-tuned to classify the watermark key set correctly. In doing so, the boundary of the decision region is stitched around the watermark key images (as



(a) Original model (b) Watermarked Model

Figure 2.7: Illustration of the approach proposed by Merrer at al. [6] for a binary classifier. The proposed algorithm first computes "true" (R and B) and "false" adversarial examples (R^{-} and B^{-}) for both classes of the model. (b) Then the model is fine-tuned such that these inputs are classified correctly. This process resembles "stitching" around data points.

shown in Figure 2.7), making the marked model distinct from similar non-watermarked models. The model is queried with the watermark key images in the detection phase, and a statistical null-hypothesis test is performed to verify the presence of the watermark. Their approach relies on the transferability of adversarial examples, in that, it is assumed that all non-watermarked models will misclassify adversarial examples in the watermark key set unless the model has been trained to remember their correct labeling. However, adversarial examples are known to be unstable, in that, not all adversarial samples crafted for one model will be misclassified by another. This may result in a high false-positive rate for the proposed approach.

It is often the case that during the embedding of the watermark, the training process of the DNN is altered, causing performance degradation. To address this drawback, several studies have proposed *passive* techniques that extract certain persistent and identifying features from proprietary pre-trained models instead of actively adding authorship information into them. IPGaurd [70] is an example of such a technique in which the model's predicated label for data

points alongside its classification boundary is used to identify pirated instances. with this approach, if a suspicious model predicts the same labels for most boundary data points, it is determined to be a stolen copy of the original model. Zhao et al. [71] proposed AFA, which utilizes adversarial examples extracted from the original model to identify pirated copies. A suspected model is considered a pirated copy if it agrees with the original model on the labels assigned to the majority of adversarial examples. However, this method can result in a high false-positive rate, in that, two models may have the same response to adversarial examples, but be totally irrelevant. To address this drawback, Wang et al. [72] proposed restricting adversarial perturbation to only high-frequency components to reduce their transferability to irrelevant (non-watermarked) models. The authors' approach is inspired by the finding in [73, 74] showing that low-frequency components in adversarial perturbations are primarily responsible for their high transferability.

We should note that passive methods such as [70, 71, 72] are vulnerable to the threat of fraudulent claims as an attacker can also extract another set of similar features from the model and use them to falsely claim ownership over the stolen model. Establishing ownership with these techniques would not be credible as they fail to associate the features extracted from the model with the vendor's identity.

Recently, Lou et al. [75] presented a method for embedding a watermark into model architecture leveraging neural architecture search (NAS) [75]. In the proposed approach, the watermark information is translated into a set of constraints on the search space of NAS, and then, the restricted search space is explored in order to find an architecture that can retain the model's performance. As the watermark information is encoded inside the architecture of the model, white-box access is necessary for verification. However, the authors demonstrated that model architecture could be easily reverse-engineered using side-channel-based methods, which

makes the approach compatible with black-box verification.

Chapter 3: Polymorphic Gate Based Watermarking Technique for Integrated Circuits

3.1 Introduction

In this chapter, we present a novel watermarking scheme for silicon IPs. In the proposed method, the watermark is embedded by selectively replacing standard library cells in the gate-level netlist of the design with polymorphic gates. Polymorphic gates are re-configurable logics whose functionality varies with changes in their execution environment such as temperature, supply voltage, or external control signals. With the proposed watermarking method, the circuit provides the correct functionality under normal operating conditions. However, when the watermark is necessary to be demonstrated, the circuit is heated to activate the polymorphic gates' control signals. This would change the circuit's functionality (observed on its primary outputs) and reveal the vendor's watermark.

Designing polymorphic gates is extremely challenging, as they exhibit irregular structures and cannot be designed using conventional gate design rules. To take on this challenge and find polymorphic gates suitable for our watermarking method, we take an evolutionary design approach and optimize the well-known genetic algorithm to implement an automatic design tool with which we can successfully construct various polymorphic gates.

Our contributions in this study are specified as follows:

- After making a second visit to the design approaches of polymorphic gates, we customize the genetic algorithm and develop a tool to automatically construct polymorphic gates; with this tool, we have successfully constructed four polymorphic gates which have not been reported before.
- We propose a polymorphic gate-based circuit watermarking scheme by replacing standard library cells with polymorphic gates. The scheme features easy detectability which is achieved by a justifiability and observability checking algorithm.
- We evaluate the proposed watermarking scheme based on the requirements discussed in Section 2.2 using ISCAS 85 and MCNC benchmark circuits and 0.13um SMIC technology. Experimental results show that our watermarking technique maintains the correct functionality of the protected circuit and incurs negligible performance overheads, is robust to counter-watermark attempts, and can be a reliable and credible tool to establish ownership over IPs.

The rest of this chapter is organized as follows: Section 3.2 provides the background on polymorphic gates and their design methodology. Section 3.3 describes our evolutionary approach for designing polymorphic gates and reports the polymorphic gates we have evolved with this framework. In Section 3.4, we propose our watermarking technique based on the evolved polymorphic gates. Section 3.5 evaluates our proposed scheme based on metrics and properties mentioned for watermarking in Section 2.2. Finally, Section 3.6 summarizes this chapter.

The content of this chapter is based on our publication listed as the reference [76].

3.2 Preliminaries on Polymorphic Gates

The advances in IC technology have led to demands for efficient ways to implement increasingly complex electronic systems. Multi-function or reconfigurable schemes are promising solutions to this problem as they aim to expand the potential scope and utility of electronic devices through a simplified and minimal number of components [77]. Traditional multi-functional systems are implemented by multiplexing between multiple stand-alone conventional subsystems. This straightforward approach satisfies the required functionality at the cost of larger implementation overheads.

Recent achievements in the field of digital circuit design bring another concept – so called *polymorphic gate* (or polymorphic circuit), which was introduced by A. Stoica as a novel type of reconfigurable scheme [78]. Different from the conventional reconfigurable circuits, polymorphic circuits need no reconfiguration switch, as the multiple-functionality is inherently embedded within them. In these circuits, transitions between different functionalities are triggered by changes in the execution environment such as temperature, supply voltage, etc. The NAND/NOR gate is the most famous example which has been fabricated using HP 0.5um technology [79]. This gate performs as a NAND gate when V_{dd} is 3.3V and when V_{dd} drops to 1.8V, it works as a NOR gate.

With two or more functions built-in one single compact structure, polymorphic circuits have found many applications where a few predefined functions have to be implemented and a global control signal selects the function, such as multi-functional adaptive systems [80, 81], finite impulse response filter [82], self-checking circuits [83], reduction of test vector volume [84, 85], etc. For these scenarios, two modes are supported in polymorphic circuits, where one

mode is considered the main operation mode and the other mode is only for special occasions. Notice that the special mode remains invisible when the circuits deliver normal function.

While standard logic gates adopt complementary topology, polymorphic gates employ rather unconventional structures at the transistor level. Due to their irregular topology, it is a challenge to find polymorphic gates. Over the years, the evolutionary approach [86, 87] has proved to be the most effective methodology to find polymorphic gate designs that best match the desired multi-functionality requirements [81]. These algorithms start by randomly combining transistors, evaluating and ranking them, and combining different partial solutions to eventually create a circuit that completes the needed functions. There are several studies that have reported designing various types of polymorphic gates based on the evolutionary approach [86, 88].

One of the most popular variants of the evolutionary approach is the *Genetic Algorithm* (GA). In the generalized version of GA [89], after the genotype (or gene) is mapped to an artificial system and the initial population of individual candidates is created, a generative process ranks candidate solutions based on a fitness function that incorporates the desired criteria, and selects the fittest candidates for mutation and reproducing the next generation. This process repeats until an acceptable solution is found. In our study, we customize the generic GA for a transistor-level gate design to find new polymorphic gates, as we will discuss in Section 3.3.

3.3 Designing Polymorphic Gates With Evolutionary Approach

As with prior arts [90], we find polymorphic gates by taking an evolutionary approach. More specifically, we adapt and customize the generic GA to implement an automated framework to search for polymorphic gates with our desired multi-functionality. Note that the proposed

Algorithm 1 Genetic algorithm for evolving polymorphic gates

Input:
1. population_size : number of individual candidates in each generation.
2. max_generation: maximum number of generations
3. $f < t, a, b \ldots >$: truth table of the desired gate function, where a,b.... are the input values and t is the external control signal to transform the function.
4. HD_threshold: threshold of hamming distance between the output of each individual and that of the desired gate function.
5. r: mutation rate of each generation.

Output: gate netlist with the desired polymorphic functionality

Step 1: Gene initialization
current_generation = 0;
for (i = 0; i < population_size; i++) **do**
  Initialize the genes and generate netlist[i];
end for

Step 2: Simulation and fitness evaluation
for (i = 0; i < population_size; i++) **do**
  for every t and input combination of a,b... **do**
    f' = Simulate(t,a,b,..netlist[i]);
  end for
  HD[i] = hamming distance (f, f');
end for

Step 3: Selection and reproduction
no_survived_indiv= 0;
for (i = 0; i < population_size; i++) **do**
  if (HD[i] == 0) **then**
    Output(netlist[i]);
    exit();
  else if (0 < HD(i) < HD_threshold) **then**
    survived_indiv = survived_indiv $\cup$ {netlist[i]} ;
    no_survived_indiv++;
  end if
end for
no_offsprings = $\lceil \frac{\text{population\_size}}{\text{no\_survived\_indiv}} \rceil$;
for (j = 0; j < no_survived_indiv ; j++) **do**
  for (i = 0; i < no_offsprings ; i++) **do**
    netlist[$j\times$ no_offsprings+i] = Randomly mutate r% of genes in survived_indiv[j]
  end for
end for
if (current_generation ! = max_generation) **then**
  current_generation++;
  Go to step 2 and then 3;
else
  exit();
end if

framework will only focus on polymorphic gates consisting of CMOS transistors to offer better integrability into the mainstream CMOS technology.

The search for desired polymorphic gates is an iterative process that starts with a random population of candidate designs (Step 1 in Algorithm 1). We denote each candidate design with a particular data structure that describes the connection of the source, gate, and drain terminals of each transistor in the design, as well as their width and length. This data structure, often referred to as *genotype* in the GA literature, contains the information necessary to describe a gate design and generate its netlist. Any changes to the genotype of a candidate have a direct effect on the netlist and other parameters of the design.

In the next step (Step 2 of Algorithm 1), candidates in the current population are rated according to a fitness function. The fitness function determines how closely a candidate gate matches the functionality of the desired polymorphic gate. We use the hamming distance between the output of the candidate design (across both modes of operation) and the desired polymorphic gate as the fitness function. Evaluating the fitness score for each candidate requires generating their netlist from their genotype, simulating their functionality in both modes of operation using the Hspice Simulator, and comparing their functionality to the desired output.

Next (Step 3 of Algorithm 1), among all the design candidates, the fittest are selected for reproducing the population of the next generation. Note that the next generation is produced by randomly modifying the genotype of the selected candidates. Steps 2 and 3 are repeated until the desired polymorphic gate is found or a number of generations have evolved and been investigated.

While the genotypes are randomly initialized and modified, there are some practical constraints that need to be followed:

Constraint 1: At least one terminal should be connected to power, ground, output, and inputs,

Table 3.1: Evolved polymorphic gates controlled by temperature.

Gate function	Transistor
OR (25°C) - AND (125°C)	6
NOR (25°C) - INV (125°C)	6
NAND (25°C) - INV (125°C)	6
AND (25°C)- BUF (125°C)	7

respectively.

Constraint 2: No floating nodes should appear in the circuits.

The first constraint is to ensure that each design will be a complete gate and the second constraint is to force the source, drain, and gate terminals of each transistor to be connected to a terminal of other transistors so that every terminal is in the path from power to the ground.

In our experiments, we target two-input one-output gates and consider temperature as the external control signal, which ranges from -25°C to 150°C. The 130nm SMIC technology is adopted and the supply voltage is set as 1.2V. It is important to note that polymorphic gates are technology-dependent, meaning that a gate design exhibiting polymorphic behavior in one process technology (e.g. 130nm SMIC) may not necessarily exhibit the same behavior in another technology (e.g. 90nm TSMC).

With the design tool, we have evolved four novel polymorphic gates for the first time. The function of these gates is listed in Table 3.1. Figure 3.1a presents the schematic of the polymorphic NOR/INV gate. As shown, the transistors are connected in an irregular topology and have taken unconventional parameters. The function of this gate is shown in Figure 3.1b. At room temperature, this gate is a NOR gate; when the temperature rises to 125°C, this gate inverts the second input.

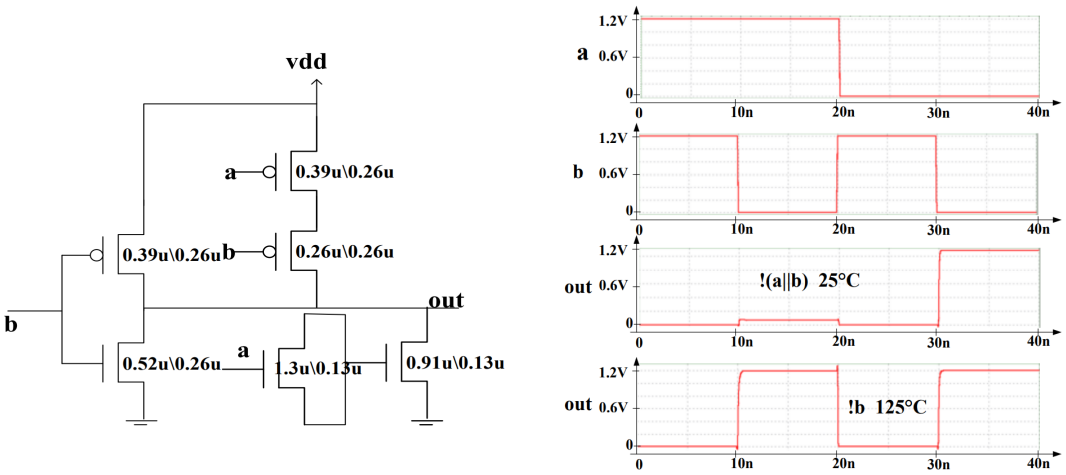


Figure 3.1: Polymorphic NOR/INV gate: (a) Topology (b) Input and output wave-forms.

3.4 The Proposed Watermarking Scheme

In this section, we propose a novel circuit watermarking scheme that relies on polymorphic gates to carry owner's signature. The proposed watermarking scheme works by replacing the standard cells in the gate-level netlist with polymorphic gates. As polymorphic gates are technology-dependent, the proposed watermarking scheme would only be appropriate for hard IPs targeting the same process technology as the polymorphic gates.

Every cell in the original netlist represents a possible location for gate replacement. However, the suitable locations for replacement should ensure: **(1)** the circuit functions correctly at normal operation mode, after the gates in these locations are replaced, and **(2)** the functionality of the modified circuit in normal mode and special mode can be differentiated by observing the primary outputs of the circuit.

To find the suitable locations to embed polymorphic gates, we address three principles that need to be followed. These principles ensure the correct functionality of the circuit and facilitate the easy detectability of the embedded watermark [91].

Principle 1: Only the gates that have the same functionality as the polymorphic gates in normal mode could be potential locations for watermark embedding.

Principle 2: There should be at least one pattern for primary inputs of the circuit that can "activate" the polymorphic gate. An input pattern can "activate" a polymorphic gate if it can set the inputs of the gate to a differentiating input value.

Differentiating Input Values (DIVs) of a polymorphic gate are the input combinations for which the polymorphic gate produces different outputs when the control signal changes. For example, input combination (1,0) is a DIV of the NOR/INV gate in Figure 3.1, since the output of this gate reverses from 0 to 1 as temperature changes from 25°C to 125°C when the two inputs are set as '1' and '0'.

Moreover, this primary input pattern should also 'propagate' the output of this gate to the primary output so that the function transformation can be observed. To find such input patterns, we adopted the *justification methodology* in VLSI testing to deduce the logic value from the input of the candidate gate backward to the primary input. Only a gate that is justifiable can be a potential replacement location.

Besides justifiability, we also need to check the *observability* of this gate location. We propagate the output of one gate by justifying the other input(s) of its successor gate as non-controlling values. Take the circuit in Figure 3.2 as an example. To propagate the output of gate 1, the first input of its successor gate 4 should be justified as '1' so that the output of gate 4 is determined by the output of gate 1. Similarly, we advance the output one gate at a time until it reaches the primary output.

Principle 3: The changes in the functionality of one polymorphic gate upon changing the temperature shouldn't influence the activation and propagation of other polymorphic gates. To

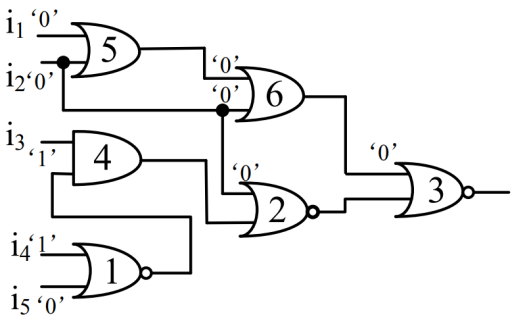


Figure 3.2: Motivation example for Algorithm 2.

cater to Principle 3, we need to incorporate the following criteria when checking the justifiability and observability of a certain location. First, according to Principle 1, every gate that has the same function as one of the polymorphic gates could be a possible location for gate replacement. When checking the justifiability or observability of a gate, we may need to justify other gates that will be replaced by polymorphic gates. For these gates, we should assign DIV with the least priority. For example, if we want to check whether a NOR gate (gate A) could be replaced by a NOR/NAND and we need to set the output of another two-input NOR gate (gate B) as '0', we will try the input patterns '11' first for B and then '01'or'10', if '11' fails. As '11' gives the same output for NOR and NAND, even if gate B is replaced by a polymorphic gate, the transition of its output when control signal changes will not influence the activation or propagation of gate A. Second, when checking the observability of a gate location we need to make sure that even if the successor gates are replaced by polymorphic gates, the gate output can still propagate in the special mode.

We give a case-by-case solution based on analyzing the truth-table of the four polymorphic gates.

Case 1: The successor gate is a two-input NOR gate. Suppose this gate is replaced by NOR/INV gate. If the second input of this gate is justified as '0' and the first input is connected

to the signal that needs to propagate, when the temperature rises, the gate inverts the second input and the output of this gate is always '1' so the propagation fails. If the two inputs are swapped (the first input is justified as '0'), the gate delivers the complementary of the second input so that the propagation can proceed. Similarly, if the successor gate is a two-input NAND (or AND) gate, it may be replaced by the NAND/INV (or AND/BUF) that inverts (or outputs) the first input. In this case, the second input should be justified as '1', and the signal to propagate is connected to the first input.

Case 2: The successor gate is a two-input OR gate. If one input of this OR gate is justified as '0' and this gate gets replaced by a polymorphic OR/AND gate, when the temperature rises, this gate turns into an AND gate. Since one input of this gate is '0', the output will be '0' no matter what value the other input (which is the signal we want to propagate) takes, which makes the propagation fail. Therefore, we will avoid propagating to two-input OR gates.

Based on these principles, we have come up with a justifiability and observability checking algorithm to find out the possible locations to insert polymorphic gates. The algorithm is specified as in Algorithm 2.

Figure 3.2 gives a motivational example of how we run this algorithm. Suppose that we choose the evolved NOR/INV gate. For the given circuit, there are three NOR gates that can be replaced. As discussed before, the two inputs of the NOR gate should be justified as '0' and '1' so that the function transition can be observed if the gate is replaced. For gate 1, i4 and i5 are set as '1' and '0', respectively. As marked in Figure 3.2, by propagating the output of g1 forward to the primary output, we derive i1= '0', i2= '0', and i3= '1'. Thus, the output of gate 1 can be observed at the primary output. Similarly, we derive 'x10xx' (x is the don't care value) for gate 3. By changing the temperature, we can observe the output to tell whether gate 1/gate 3 is

Algorithm 2 Justifiability and observability checking algorithm - Part 1

Input: Gate level netlist of the original design, polymorphic gate g_p whose function changes from f_1 to f_2 when temperature changes.
Output: locations that can be replaced by polymorphic gates

---

for (every gate **g** in the gate netlist) **do**
  if (**g** functions as $\mathbf{f_1}$ && Activate(**g**) == success && Propagate(**g**) = success) **then**
    Report **g** is a possible location to embed polymorphic gate
  end if
end for

procedure Activate (gate g)
  for (i= 0; i $<$ number of inputs for g; i++) **do**
    if (justify (g's i_{th} input, v_i) == fail) **then**
      return fail;
    end if
  end for
  return success;
end procedure

procedure Propagate (gate g)
  if (g is output) **then**
    return success;
  end if
  for (i = 0; i $<$ number of fanout gates for g; i++) **do**
    g_f = g's i_{th} fanout gate;
    n_i = number of inputs for g_f ;
    if ((g_f is NAND or g_f is AND) && n_i ==2) **then**
      if (justify (g_f's 2nd input, 1) == success) **then**
        return success;
      end if
    end if
    if (g_f is NOR n_i == 2) **then**
      if (justify (g_f's 1st input, 0) == success) **then**
        return success;
      end if
    end if
    if (n_i != 2) **then**
      for (j = 0; j $<$ n_i; j++) **do**
        if (justify (g_f's j_{th} input, v_{non_con}) == success) **then**
          return success;
        end if
      end for
    end if
  end for
  return fail;
end procedure

Algorithm 2 Justifiability and observability checking algorithm - Part 2

```
procedure Justify(gate g, justification value v)
  if (v != g's existing output value) then
    backtrace();
  else if (g functions as f_1 && v = v_d ) then
    for (every input patterns giving same output for f_1 and f_2) do
      for (i= 0; i < number of inputs for g; i++) do
        if (justify (g's i_th input, v_i') == fail) then
          break;
        end if
      end for
    end for
    for (i= 0; i < number of inputs for g; i++) do
      if (justify (g's i_th input, v_i) == fail) then
        return fail;
      end if
    end for
  end if
  ...
end procedure
```

a standard NOR gate or a polymorphic NOR/INV gate. For gate 2, such an input pattern doesn't exist, so gate 2 is not an ideal location to insert a polymorphic gate.

3.4.1 Watermark Embedding

To embed a watermark with the proposed framework, the following actions need be taken in the design phase, as illustrated in Figure 3.3. *First,* determining the locations (e.g. gate 1 and 3 in Figure 3.2) to insert polymorphic gates with Algorithm 2. The input vectors for detecting the watermark are also obtained at this step ('00110' for gate 1 and 'x10xx' for gate 3). *Second,* generating the watermark sequence. Bit '1' ('0') in this binary pattern means a standard logic gate in a certain location will (will not) be replaced by a polymorphic gate. *Third,* replacing standard logic gates with polymorphic gates. Thus, the watermark is embedded in the 'hidden' function when the circuit works at special mode.

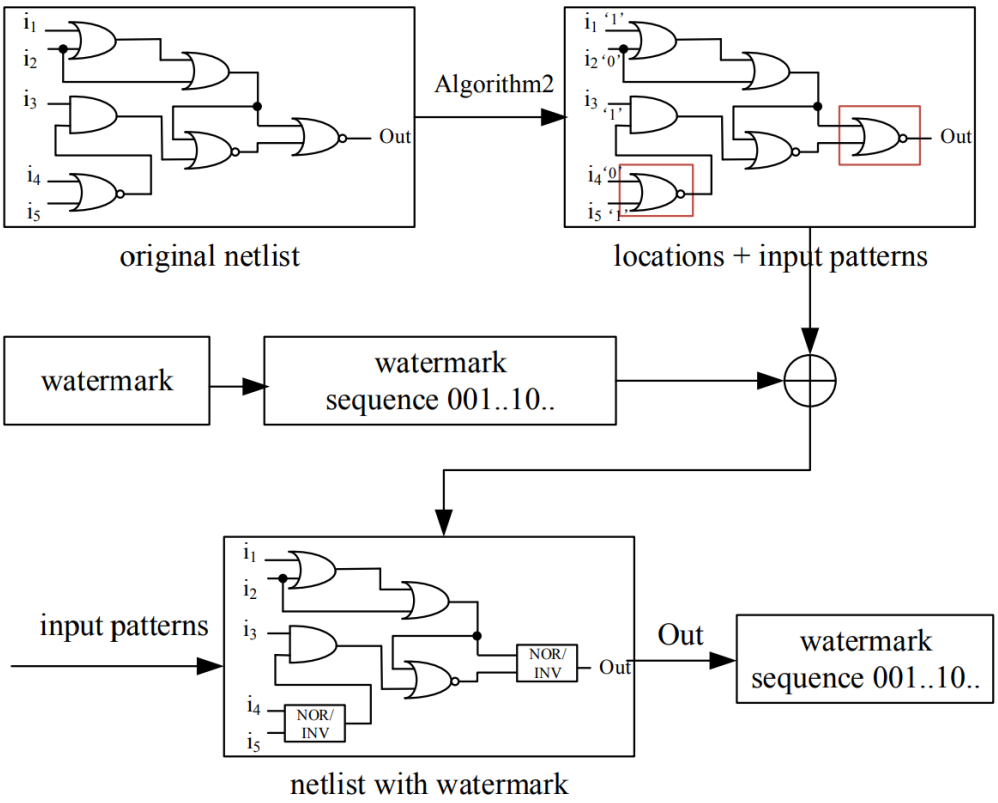


Figure 3.3: Design flow of the proposed watermarking scheme.

3.4.2 Watermark Detection

To detect the watermark, the users need to feed the input vectors provided by the designers into the circuits and compare the primary outputs gathered at normal mode and special mode. If a difference is observed at a certain primary output, it indicates that the circuit has employed a polymorphic gate at the corresponding location, which means a bit '1' of the watermark sequence is detected, and vice versa. Therefore, the watermark sequence can be retrieved which can be used to prove the ownership of the design.

3.5 Experimental Evaluation and Discussion

In our experiments, we selected several circuits from ISCAS 85 and MCNC benchmarks to perform watermarking. The circuits are described in Verilog HDL and synthesized using 130nm SMIC technology with the supply voltage set to 1.2V. For simplicity, the behavioral design is mapped to inverters and two-input/three-input/four-input NAND/NOR/OR/AND gates. The evolved gates are replaced in the gate-level netlist with a netlist modifier written in C.

The timing and power of the polymorphic gates are measured using Hspice and their layout area is estimated based on the transistor parameters. With these measurements, we incorporate the polymorphic gates into the SMIC 0.13um synthesis library as standard cells. We measure the area, delay, and power of the original netlist and the modified netlist after polymorphic gates are embedded using Synopsys Design Compiler.

In our experiments, we embed watermarks of varying sizes into each benchmark. We start from the simplest case where only a small watermark of 4 bits is needed. For that, we randomly choose 4 possible gate locations to embed polymorphic gates. We traverse all 16 cases where the watermark value ranges from "0000" to "1111". For each watermark sequence, we replace the gate at a certain location with a polymorphic gate if the corresponding bit in the sequence is '1'. Therefore, we obtain 16 netlists embedded with 16 different watermarks and measure their maximum path delay, area, and power using the Design Compiler.

Next, we consider scenarios similar to real-world practices where a large payload for the watermark is needed. For that, we consider embedding watermarks of 20 and 30 bits into each benchmark. Since it is not practical to exhaust all the possible 2^{20} or 2^{30} cases, we randomly generate 100 20-bit (30-bit) watermark sequences, embed the polymorphic gates correspondingly,

Table 3.2: Number of locations that a polymorphic gate can be inserted to embed a watermark.

Circuit	Number of all gates	Number of possible locations
C880	290	94
C1355	424	32
C1908	396	114
C3540	943	83
C5315	1428	556
dalu	1228	131
des	3483	1084
ex5	609	48
i8	1174	277
i10	1914	526
vda	483	131

and obtain 100 modified netlists. The average of maximum path delay, area, and power of these netlists is also measured.

In what follows, we evaluate the performance of our watermarking scheme in relation to the properties mentioned in Section 2.2.

Capacity: A watermarking scheme is required to embed enough information to support the vendor's ownership claim in front of a court of law. Using Algorithm 2, for each benchmark circuit, we identified all locations (gates) that can be replaced with polymorphic gates to embed a watermark. Taking a look at Table 3.2, every benchmark circuit has ample places to insert polymorphic gates in. This means that our watermarking scheme can embed large multi-bit signatures and offers high embedding capacity, making it ideal for proof-of-authorship applications.

Fidelity: A watermark embedded into an IP must not impact its functionality and should have little or no effect on its performance according to the fidelity requirement. The proposed watermarking method inserts the polymorphic gates in locations that ensure the circuit functions

Table 3.3: Area, performance and power overhead of the proposed scheme for watermarks of varying sizes.

Circuit	4 bit watermark			30 bit watermark			20 bit watermark		
	Δ Delay (%)	Δ Area (%)	Δ Power (%)	Δ Delay (%)	Δ Area (%)	Δ Power (%)	Δ Delay (%)	Δ Area (%)	Δ Power (%)
C880	0	0.66	0.43	7.88	4.05	5.54	0.54	2.77	3.97
C1355	6.62	0.47	0.24	12.34	3.46	0.89	11.44	2.22	0.70
C1908	1.70	0.12	1.11	4.62	4.15	0.47	6.08	2.52	0.76
C3540	0.18	0.19	0.47	7.24	1.53	1.44	8.17	0.93	0.48
C5315	0	0.50	0.46	0	0.47	0.75	0	0.14	0.61
dalu	0	0.23	0.44	0	1.78	0.49	0.47	1.17	0.14
des	0	0.07	0.53	0	0.48	0.44	0.28	0.28	0.49
ex5	0	0.37	0.54	5.96	2.44	1.33	0	1.63	1.40
i8	0	0.16	0.51	0.30	1.21	0.67	0.30	0.83	0.58
i10	0.74	0.09	0.27	1.79	0.71	0.43	3.13	0.47	0.26
vda	0	0.35	0.006	4.96	2.59	0.42	3.19	1.97	0.32
Avg	0.84	0.29	0.45	4.10	2.08	1.17	3.05	1.36	0.88

correctly under normal conditions. Therefore, the functionality of the marked and original design would be the same under normal conditions.

The performance of circuits is often measured through factors such as maximum path delay, area, and power consumption. Table 3.3 shows how embedding watermarks of various sizes impacts these metrics for each benchmark circuit. From Table 3.3, we can see that a small size watermark (4-bit watermark sequence) introduces very little overhead, which is an average of less than 1% increase in delay, area, and power consumption, respectively. Additionally, for larger watermark payloads, the watermarking scheme incurs only less than 5% overhead on average. It is evident from the results that the proposed watermarking scheme not only maintain the circuit's correct functionality but also has a negligible effect on its performance.

Robustness: Robustness requires the embedded watermark to be resilient to both unintentional and malicious changes in the watermarked work. As discussed, the proposed watermarking technique is applicable for hard IPs which are delivered in the form of a final physical layout optimized for a target process technology. In this context, to remove the watermark, the adversary must perform certain tasks: (1) reverse-engineering the IP netlist from its physical layout (2)

identifying inserted polymorphic gates and their multi-functionality, and (3) then replacing them with standard cells whose functionality is identical to that of the polymorphic gates in their normal mode of operation. Even if we assume a strong adversary who is capable of performing tasks 1 and 2, it would be extremely hard if not impossible for them to remove and add cells to hard IPs without compromising the performance of the IP or disrupting its correct functionality.

Credibility: Credibility implies that no adversary should be able to forge a secondary watermark or find an accidental watermark in the circuit. In the proposed method, forging a new watermark poses the same challenges as removing the existing one, as both require major modifications to a well-formed hard IP. Moreover, finding an accidental watermark necessitates finding polymorphic gates in the design which do not play a role in carrying the watermark embedded by the legitimate owner. This is not possible as every polymorphic gate in the design is intentionally inserted to carry the owner's signature.

Efficiency: The embedding costs in the proposed scheme include the timing overhead of finding polymorphic gates for the target CMOS technology and finding possible locations to insert polymorphic gates. Our experiments confirmed that the embedding cost for any of the selected benchmarks was within hours which shows that the proposed watermarking scheme will have no noticeable impact on the time-to-market of the protected circuits. Moreover, the detection cost of our watermarking scheme is also negligible as it is simply a matter of heating the circuit and providing a set of input cases to the circuit. Therefore, the proposed watermarking scheme satisfies the efficiency requirement.

Reliability: The proposed watermarking scheme is reliable as it guarantees a zero false-positive rate. This is because unwatermarked circuits (i.e. circuits that do not contain any polymorphic gates) do not exhibit different functionalities after changes in their operating environment.

3.6 Summary

In this chapter, we proposed a watermarking scheme for ICs which can work as an enabling technology to establish ownership over IPs and protect legal originators against piracy and copyright violation. The proposed scheme is based on embedding polymorphic gates that have been evolved with the genetic algorithm into the gate-level netlist. Experiment results demonstrate that our scheme incurs path delay, area, and power overheads, can embed large multi-bit watermarks and yields a zero false-positive rate, making it ideal for copyright protection applications.

Chapter 4: GradSigns: A Robust Watermarking Framework for Deep Neural Networks

4.1 Introduction

In this chapter, inspired by the constraint-based watermarking paradigm for IC designs [92, 93], we introduce GradSigns, a novel watermarking framework for classification DNNs. GradSigns embeds the watermark information by imposing a statistical bias on the expected gradients of the cross-entropy cost function with respect to the model's input. Notably, GradSigns has little or no impact on the prediction accuracy of the marked model and can be verified over API access. More importantly, unlike contemporary black-box watermarking methods, GradSigns is able to embed more than one bit of watermark information into DNNs and is also shown to be extremely robust against general counter-watermark methods (such as parameter pruning, model fine-tuning, and query invalidation), and possible adaptive watermark removal techniques which are designed with complete knowledge of our framework.

Our main contributions and findings are as follows:

- We propose GradSigns, a novel and robust watermarking framework for classification DNNs. To the best of our knowledge, GradSigns is the first watermarking method leveraging the gradient of the model's components to embed the owner's signature.

- We evaluate the performance of GradSigns on DNNs trained for three different image classification tasks including object and hand-written digit classification and face recognition. Our evaluations reveal that GradSigns is able to successfully embed watermark information with negligible impact on the model's performance. We also show that GradSigns is highly effective and reliable in protecting the owner's copyright.

- We demonstrate, thorough rigorous experiments, that unlike contemporary black-box watermarking methods, our framework can embed a large amount of information into DNNs, and watermarks embedded by GradSigns are robust against known and proposed adaptive counter watermark attacks mounted by strong adversaries.

The content of this chapter is based on our publication listed as the reference [94].

4.2 Threat Model

Our threat model includes two parties: the *model vendor* and the *adversary*. The model vendor owns model M, a DNN which they have engineered and trained for a certain task using the dataset D. Dataset D is collected and owned by the model vendor. The second party, the adversary, is an entity that doesn't have the required resources for designing and training a top-notch model, and wishes to make a profit out of model M without paying any copyright fee to the model vendor. The adversary can be a company that has purchased the license of M for one of their products and want to deploy it on another one without paying additional copyright fees. They can also be any entity who somehow has got their hands on the model, and wish to sell it on the darknet. The model vendor's goal is to protect model M against IP infringements by means that enable them to identify and prove their ownership over the pirated version of M. On the

other hand, the adversary's ultimate goal is to continue profiting from M without getting caught by law enforcement.

In our threat model, we assume the strongest adversary who has the expertise and the computation power required for training a model. However, the dataset that they have available for the learning task is far smaller than the dataset D owned by the model vendor, and therefore is not large enough for them to train a top-grade model from scratch. If the adversary had access to dataset D, they would not need to hijack M, as they are capable of training the model themselves. Similar to prior arts, we assume that the adversary is capable of trying any of the general anti-watermark attacks such as parameter pruning, model fine-tuning, and query invalidation and modification to remove the watermark or obstruct its verification. In addition, we assume that the adversary is aware of all existing watermarking techniques, and is capable of designing adaptive anti-watermark schemes to target the deployed method. Due to such possible counter-watermark attempts, it is safe to assume that a hijacked model will go under modifications before being monetized by the adversary. The adversary accomplishes their goal, if they can remove the vendor's signature or obstruct watermark verification without sacrificing too much on the performance of the model. Note that a counter watermark attempt that drastically degrades the model's performance is not considered successful.

4.3 The Proposed Watermarking Scheme

The foundation of our proposed watermarking framework, GradSigns, is based on the fact that there can be more than one unique solution to the non-convex optimization problems that deep learning models are designed to solve. Deep learning models, mainly due to their

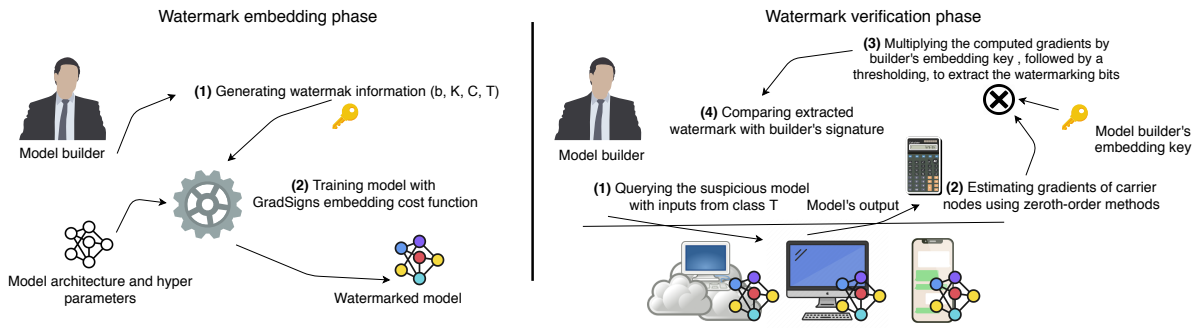


Figure 4.1: Workflow of watermark embedding and verification using GradSigns.

over-parametrization, transform the loss surface to have a manifold of optimal solutions which enables optimization algorithms to find multiple solutions yielding comparable performances on the testing data set [95, 96]. For classification tasks, each solution corresponds to a unique decision boundary with similar classification accuracy.

Intuitively, GradSigns finds a solution (i.e. a set of model parameters) corresponding to a decision boundary that not only results in comparable performance on the original task but also fulfills an additional goal which is carrying the owner's watermark information. GradSigns works by embedding watermark information into the expected gradient of the cross-entropy cost function with respect to the model's input. For any input sample x, the gradient of the cost function with respect to the input is a vector tangent to the model's cost function surface, and perpendicular to the decision boundary at point x. Therefore, by imposing a statistical bias on these gradients, GradSigns is essentially reshaping the decision boundary to incorporate the desired watermark information. For simplicity, we refer to the gradient of the cross-entropy cost function with respect to the input of the model as *gradient of input* or *input gradient* in the rest of the chapter. Figure 4.1 illustrates the workflow of GradSigns, and Section 4.3.1 and 4.3.2 describe the watermark embedding and verification phase of our watermarking framework.

4.3.1 Watermark Embedding

Forcing a random statistical bias on the gradient of inputs to the model can drastically degrade its performance on the classification task. To ensure a successful marking without sacrificing the model's performance, the watermark needs to be embedded while optimizing the model for the original task. For that reason, similar to prior arts such as [54, 57], we embed the watermark into the host model by including a regularizer term in the model's training cost function. The final training cost function including the regularizer term is defined as:

$$J(x|\,\theta,\,y) = J_{cross-entropy}(x|\,\theta,\,y) + \lambda\, J_{Embedding}(x|\,\theta,\,y) \tag{4.1}$$

Here, θ denotes the model's parameters, x is an input sample, y is the ground truth label for input sample x, $J_{cross-entropy}(.)$ is the task-specific cost function which is the cross-entropy function for classification problems, λ is the trade-off hyperparameter, and $J_{Embedding}(.)$ is the watermark embedding regularizer term which penalizes the distance between the expected value of input gradients and the desired watermark. Before defining the embedding regularizer term, we explain the steps that the model vendor needs to take prior to embedding the watermark. These steps are as follows:

Step 1: Generating an N-bit vector $b \in \{0,1\}^N$ to be used as the watermark.

Step 2: Randomly selecting a set C of input neurons to carry the watermark. We refer to set C as the *watermark carrier set*, and to neurons in C as *carrier nodes*. The gradient of inputs observed on neurons in the career set participate in embedding the watermark.

Step 3: Generating an *embedding key* $K^{N\times|C|} \in [-1,1]^{N\times|C|}$. Embedding key is a transformation

matrix that maps the expected gradient of carrier nodes to a binary vector of size N.

Step 4: Selecting a random target class T. Images from class T are used to calculate the gradients of carrier nodes.

Note that generating the watermark b and embedding key K can either be done randomly by using a Random Number Generator (RNG) or by hashing a message containing information that can be used to prove the vendor's ownership as further explained in Section 4.4.4. In our method, the watermark is successfully embedded if the following property holds:

$$\forall j \in \{0, 1, ..., N-1\}, \quad \chi_{[0,\infty)} \left(\sum_{i=0}^{|C|-1} K_{ji} G_i \right) = b_j \tag{4.2}$$

Here, $G \in \mathbb{R}^{|C|}$ is the expected gradient of cross-entropy function with respect to carrier nodes in C, measured over a sample of images from target class T, K is the model vendor's embedding key, b_j is j_{th} watermark bit, and χ is a step function outputting one for values greater than zero.

For each watermark bit j, Equation 4.2 denotes a linear inequality where the expected gradients of carrier nodes (G) are the variables, and the j-th row of the embedding key (K) are the coefficients, as shown in Equation 4.3. This linear inequality is essentially denoting a half-space where acceptable values of expected gradients can reside for a successful embedding of watermark bit j.

$$(-1)^{b_j} \sum_{i=0}^{|C|-1} K_{ji} G_i < 0 \tag{4.3}$$

The task of embedding each watermark bit j can also be viewed as a binary classification task with a single layer perceptron (SLP) where the parameters of the perceptron layer are fixed to a constant value equal to j_{th} row of the embedding key (K), and only the input of the SLP, i.e. the gradient of the carrier nodes, are being trained. To this end, we utilize a binary cross-entropy loss

function in the embedding regularizer to embed each watermark bit:

$$J_{Embedding}(\theta) = -\sum_{j=0}^{N-1}(b_j\, log(y_j) + (1 - b_j)\, log(1 - y_j)) \tag{4.4}$$

Here, $y_j = \sigma(\sum_i K_{ji}G_i)$ is the output of the SLP corresponding to the j_{th} watermarking bit, and σ is the sigmoid function.

4.3.2 Watermark Extraction

To extract a watermark embedded by GradSigns, the first step that the model owner needs to take is computing the expected gradient of the carrier nodes. In the white-box setting where the owner has access to the internal configurations of the suspicious model, the gradients can be calculated by backpropagation. However, in the black-box setting, computing gradients via backpropagation is not possible.

To enable watermark extraction in the black-box setting, we propose using a zeroth-order gradient estimation method to calculate the expected gradients of the carrier nodes. Zeroth order methods can estimate gradient with respect to any direction v by evaluating the cost function value at two very close points located along this direction [97, 98]. We use the difference quotient to estimate the gradient of the cost function with respect to carrier nodes as shown below:

$$\hat{G}_c(x) \;=\; \frac{\partial\, J(x)}{\partial x_c} \approx \frac{J(x \,+\, he_c) \,-\, J(x\,)}{h} \tag{4.5}$$

where $\hat{G}_c(x)$ is the estimated gradient of carrier node c at point x, h is the estimation step length which is set to 0.0001 throughout our experiments, e_c is a standard basis vector with 1 at the

component corresponding to career node c, and 0s elsewhere. Please note that the value of cross entropy function $J(x)$ can be computed in the black-box setting, given the model's output and the ground truth label for input x.

In Equation 4.5, for each input x, the gradient of a carrier node is calculated by evaluating the value of the cost function for two points whose coordinates are the same except for the one coordinate corresponding to the carrier node. The gradient estimation error of career nodes, not including the error introduced by limited numerical precision, is in the order of $O(|C| \cdot h^2)$ [99]. For any input x, we need to evaluate the cost function $|C| + 1$ times to estimate the gradients of all carrier nodes. We note that our watermark extraction naturally applies to more query-efficient gradient estimation methods such as [100].

After calculating the expected gradients for all carrier nodes, the model vendor can retrieve the embedded watermark by multiplying the expected gradients with their embedding key. The model belongs to the vendor, if the Bit Error Rate (BER) of the extracted watermark is lower than a certain threshold. In Section 4.4.2, we explain how the BER threshold is determined to assure a reliable watermark verification, i.e. low false positive and high detection rates.

4.4 Experimental Evaluation and Discussion

In this section, we evaluate performance of GradSigns with respect to the requirements mentioned in Section 2.2 on benchmark models trained on various image datasets including CIFAR-10 [101], SVHN [102], and YTF [103]. The CIFAR-10 dataset [101] is an object classification dataset containing 50,000 training and 10,000 testing samples belonging to 10 different classes. The DNN model used for this dataset is ResNet20 [104]. YouTube Face (YTF) [103] is a face

recognition dataset containing images of 1,595 individuals captured from videos on YouTube. We retrieve 120,000 images of 1200 individuals (100 images per individual) and use 75% of the images for training and the remaining as the test set. In our experiments, each image has been resized to $40 \times 40 \times 3$ dimension. SVHN [102] is a dataset of more than 100k images of digits cropped out of images of houses and street numbers. The dimension of each image in this dataset is $32 \times 32 \times 3$.

The ResNet20 [104] and Deep-ID [105] are benchmark neural networks considered for CIFAR10 and YTF datasets, respectively. For the SVHN dataset, a convolutional neural network with 6 convolution layers and 2 fully connected layers is selected as the benchmark model.

We embed watermarks with three different sizes of 16, 32, and 64 bits using GradSigns for all benchmark models. Images of airplanes, number "1", and individual number 1 are used as the target class for CIFAR-10, SVHN, and YTF benchmarks models, respectively. Trade-off hyperparameter λ is empirically set to a value in the range $(0.0, 1.0]$ depending on the benchmark, size of the watermark carrier set $|c|$, and watermark length. Selecting a large λ can degrade watermarked model's performance while choosing a small λ leads to failure in successfully embedding the watermark. In our experiments, λ is set to the smallest value (over trials with varying values for λ) that can successfully embed the watermark. Moreover, our experiment showed that using larger carrier sets increases the Bit Embedding Success Rate (BESR) of embedding attempts. BESR denotes the ratio of watermark bits that are inserted successfully into the model. However, we note that a large carrier set would translate to higher watermark verification costs in the black-box setting, which will be further discussed in Section 4.4.5. To this end, in our experiments, we select the smallest watermark carrier set size (over trials with varying carrier set sizes) which is capable of embedding the watermark successfully. Table 4.6

reports the chosen watermark carrier set sizes for all benchmarks.

In what follows, we examine the properties of an effective watermarking method as summarized in Section 2.2.

4.4.1 Fidelity

Fidelity requires the watermarking method to have minimal impact on the functionality and the performance of the host model. For classification DNNs, performance and functionality are similar concepts both of which describe the accuracy of the model on testing (unseen) samples.

Comparing the prediction accuracy of the baseline (unmarked) and marked models in Table 4.1 demonstrates that GradSigns meets the fidelity requirement as it incurs on average less than 1% drop in the performance of all benchmark models.

SVHN and YTF baseline benchmarks are trained using the same training setting (epochs, learning rate, hyperparameters, etc.) adapted for their watermarked counterparts, and their best performance on validation sets is reported in Table 4.1. For the CIFAR10 baseline model, we are reporting the maximum validation accuracy claimed in the original paper [104] for a ResNet20 trained on the CIFAR10 dataset. GradSigns is compliant with this requirement for (1) it embeds the watermark information by simultaneously optimizing for both model's prediction accuracy and the watermark embedding regularizer mentioned in Section 4.3.1, and (2) it only affects parts of the decision boundary that pertain to samples of the target class.

4.4.2 Reliability

There is a possibility that the watermark extracted from similar unmarked models partly matches the owner's signature. Therefore, to prevent false positives (false alarms) in watermark

Table 4.1: Test accuracy of watermarked models. GradSigns has a negligible impact on the performance of marked models.

Dataset	Baseline Acc.	**GS-16**	**GS-32**	**GS-64**
CIFAR-10	91.2%	90.1%	90.4%	90.5%
SVHN	96.1%	95.5%	95.7%	95.3%
YTF	99.6%	99.6%	99.6%	98.6%

detection, it is crucial to establish a threshold for tolerated mismatched bits on the extracted watermark. Based on this threshold, the model owner can decide and prove whether the model under investigation belonged to them or not.

We rely on null hypothesis testing for verifying the ownership of a model. As mentioned in Section 4.3.1, possible values of expected gradients for a successful embedding of a *single* watermark bit form a half-space. Therefore, the probability that expected gradients of a null-model, any unmarked model trained for the same task, coincidentally reside in this half-space is $\frac{1}{2}$. Let η be the maximum number of bit errors tolerated on the extracted watermark to safely claim the model's ownership. Assuming that model M is a null-model, the probability that the extracted watermark from M has at most η erroneous bits is as follows:

$$P(n_{error} \leq \eta | M) = \sum_{k=0}^{\eta} \binom{N}{k} (\frac{1}{2})^{N-k} (1 - \frac{1}{2})^{k} \tag{4.6}$$

Here, n_{error} is a random variable denoting the number of erroneous bits, and N is the length of the embedded watermark. To reliably reject that a model is behaving like a null model, we require the probability in Equation 4.6 to be upper bounded by τ, a close to zero threshold. Solving $P(n_{error} \leq \eta | M) < \tau$ for η yields the maximum number of bit errors tolerated on the extracted watermark to safely verify the model's ownership. In all our experiments, we set the threshold τ to 3×10^{-3}. With $p - value < 3 \times 10^{-3}$, the minimum number of correct (matching) bits

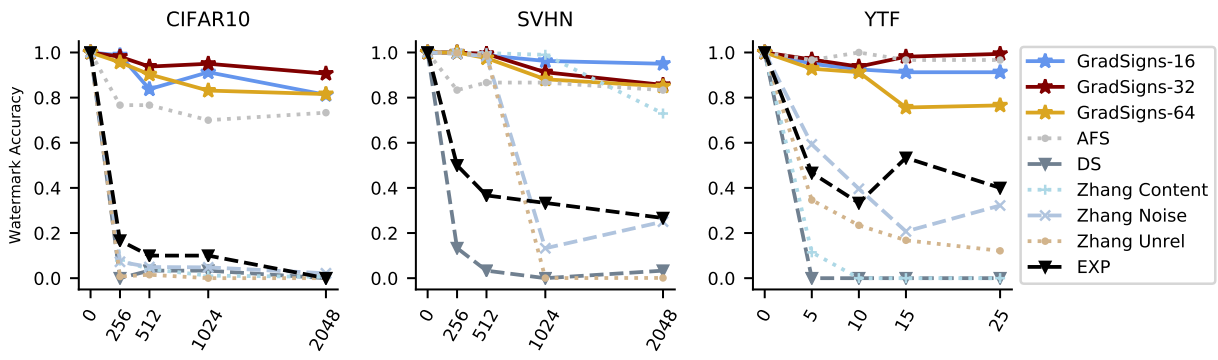


Figure 4.2: Watermark accuracy of contemporary watermarking methods in presence of model pruning and fine-tuning removal attacks. The horizontal axis of each diagram demonstrates the number of samples available for each classification label in the adversary's dataset.

required to reliably claim ownership of a model marked with signatures of size 16, 32, 64 bits are 14, 25, and 44 bits, respectively. A watermark is declared *existent and verifiable* in a model *if and only if* the extracted signature has fewer erroneous bits than the computed threshold η.

4.4.3 Robustness

In this section, we evaluate the robustness of our framework against various counter watermark schemes. We start by considering general counter watermark attacks such as *model fine-tuning*, *model pruning*, and *query invalidation-modification*, which are applicable to any watermarking framework, and have been widely considered in prior arts. Additionally, we perform a thorough comparison of resiliency of GradSigns and other contemporary black-box watermarking methods including DS [57], EXP [65], AFS [6], and Zhang's [1] against such counter watermark attacks. For a fair evaluation, we don't experiment with white-box techniques such as Uchida's [54] or white-box DeepSigns [57] as they benefit from stronger underlying assumptions.

We also assume that the adversary is aware of our framework, and consider several adaptive counter watermark attacks to further solidify the evaluation of GradSign's robustness.

Parameter pruning and model fine-tuning are two of the most common attacks against watermarks in literature. Fine-tuning is retraining the model's parameters with the training or a new dataset to find other local minima with better performance on the original task. Parameter pruning is a compression technique used to reduce the model's computational complexity and memory usage. It is often applied to DNN models deployed on embedded systems and mobile devices. Similar to prior arts, we utilize the parameter pruning technique in [66] where $p\%$ of model parameters with the smallest absolute values are set to zero and removed from the computation graph of the model. p is the *pruning rate* which decides the portion of model parameters to be removed. Parameter pruning is usually followed by model fine-tuning so the model can recover from the possible performance drop due to model compression.

Figure 4.2 shows the watermark accuracy of all seven watermarking methods in presence of model pruning and fine-tuning attacks across all three benchmarks. For methods AFS[6], DS[57], and EXP[65], we have embedded 30 key samples by fine-tuning the host model as instructed in the original work. Note that all the parameters regarding the watermark embedding procedure were adopted from the original papers. The details of watermark embedding using Zhang[1] are reported in Table 4.2. The methods proposed in [62, 106] are not evaluated here as they essentially follow a similar approach as [1] to embed the watermark, and their contribution is not introducing a new watermarking technique but establishing a reliable watermark verification using cryptography.

The horizontal axis of diagrams in Figure 4.2 *denote the number of samples available per classification label in the adversary's dataset.* For each benchmark, we have considered adversaries with four different sizes of sub-datasets in hand. Stronger adversaries have access to datasets of comparable size to the original training dataset while weaker ones have only a

Table 4.2: Implementation details of Zhang et al. [1] method.

Dataset	Watermark type	Key samples	# of key samples	Label for key samples
CIFAR-10	Content	CIFAR-10 airplanes superimposed with string "TEST" in gray	5000	cars
	Unrelated	MNIST[107] "1"	6000	cars
	Noise	CIFAR-10 random superimposed with Gaussian noise $\mathcal{N}(0,\ 0.005)$	3000	cars
SVHN	Content	SVHN "4" superimposed with string "TEST" in gray	5000	SVHN "1"
	Unrelated	CIFAR-10 cars	5000	SVHN "1"
	Noise	SVHN random superimposed with Gaussian noise $\mathcal{N}(0,\ 0.005)$	1000	SVHN "1"
YTF	Content	YTF individual #4 superimposed with string "TEST" in gray	81	YTF individual #1
	Unrelated	CIFAR-10 deer	5000	YTF individual #1
	Noise	YTF random superimposed with Gaussian noise $\mathcal{N}(0,\ 0.005)$	1000	YTF individual #1

small number of samples to work with. We note that an adversary with access to datasets larger than what's considered in Figure 4.2 has no incentive to hijack the model as they are capable of training the model from scratch with comparable performances. As shown in Figure 4.2, ***only*** watermarks embedded by GradSigns show **consistent** resiliency against "pruning+fine-tuning" attack, and remain existent and verifiable across all three benchmarks.

In our experiments, 70% of adversary's datasets is dedicated as the training set, and the rest as the validation set. Every watermarked model is pruned with 10 different pruning rates varying from 0% to 90% resulting in 10 different compressed models. Then, each pruned model is fine-tuned for 10 epochs with early stopping on the accuracy of validation. The learning rate is set 0.0005 which is equal to or greater than the learning rate in the final stage of training of benchmarks. The watermark accuracy reported in Figure 4.2 is the minimum watermark accuracy observed across all the "pruned+fine-tuned" models whose test accuracy doesn't fall beneath 10% of the baseline model's. As mentioned in Section 4.2, a removal attempt is not successful if it leads to a significant drop in the performance of the model.

Query invalidation-modification [65] is a counter watermark attack in which the adversary utilizes an auto-encoder to identify and modify model queries that are executed to extract the

watermark. Namba et al. [65] argue that if an input x is drawn from the same distribution as the training data, the reconstruction loss of input x introduced by any auto encoder AE trained on the same dataset should be relatively smaller compared to an input sample coming from a different distribution (e.g. a sample created by superimposing images with noise or a context). In this approach, besides the reconstruction loss introduced by the auto-encoder, the Jensen-Shanon divergence between predicted class probabilities of the model for input x and the reconstructed input $AE(x)$ is used as a measure to differentiate ordinary input samples from watermark key set. This distance is larger for key samples compared to ordinary inputs, as most existing watermarking methods label watermark key samples differently from their original classes. In this approach, if input sample x is identified as suspicious, i.e. possibly belonging to the vendor's watermark key set, the model will be queried using $AE(x)$ instead. While this counter watermark attack has proven effective [65] against watermarking techniques proposed by [1] and [57], it doesn't apply to GradSigns. The watermark key set for GradSigns is constructed from samples of training data without any modification or relabeling, which renders this attack futile against our method.

Assuming that the adversary is aware of our framework, they might attempt to corrupt the estimated gradient for carrier nodes to prevent watermark verification. To achieve this goal, the adversary can take any of the following approaches: (a) *Model tampering*: modifying the model parameters to change values of gradients (b) *Input tampering*: modifying inputs to the model to incur error on the estimated gradients, and (c) *Output tampering*: tampering reported prediction scores to corrupt approximated gradients.

Model fine-tuning and model pruning are known instances of model tampering attacks that have been thoroughly investigated in this section. Besides these counter watermark attacks, we

consider *adversarial fine-tuning* and *model quantization* in our experiments which fall within the same category of model tampering attack. To further broaden the scope of our evaluations, we propose three adaptive counter watermark attacks against GradSigns, namely *input tampering*, *score rounding*, and *score perturbation*, which are instances of input and output tampering attacks.

For the rest of this section, we only report experimental results for 64-bits marked benchmarks. However, we note that results for other benchmarks are consistent with the statements we make in this section.

Model Quantization is a compression technique to decrease the model's memory footprint by reducing the number of bits that are required to represent its parameters. Along with weight pruning, model quantization is among the popular techniques for making inference more efficient in resource-limited settings. We conjectured that model quantization might tamper with the information carried over the gradients by modifying model weights, however, our experiments showed the contrary. In this experiment, we first quantized parameters of marked benchmarks to 8 fixed point integers, and then performed watermark extraction. We were able to verify the watermark successfully from all benchmarks, which showed model quantization can not remove watermarks embedded by GradSigns. Table 4.3 reports the performance overhead of model quantization across benchmarks and the BER of the extracted watermark from each quantized model. Note that for all benchmarks, the reported BER is well below the tolerable threshold determined in Section 4.4.2, meaning that watermarks can be verified successfully.

Table 4.3: Robustness of GradSigns against model quantization.

	CIFAR10	YTF	SVHN
BER (# bit error/total)	2/64	1/64	0/64
Performance difference	-1.42%	-1.73%	-0.6%

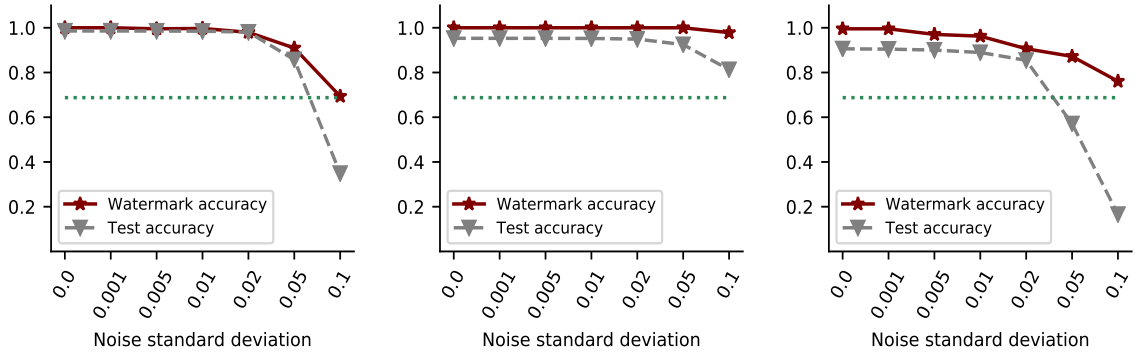
Adversarial Fine-tuning. The adversary might perform additional fine-tuning with non-standard objectives to more directly attack the gradient. To this end, we considered fine-tuning benchmark models with adversarial examples which is normally performed for improving the robustness of DNNs against test time adversarial attacks. In this experiment, a set of adversarial examples are generated following the FGSM method [18] and benchmark models are fine-tuned for another 5 epochs with a mixture of generated adversarial examples (with their correct label) and original training samples. As reported in Table 4.4, our evaluations showed that watermarks embedded by GradSign remain verifiable after adversarial fine-tuning. Adversarial training resulted in the worst case 5 mismatched bits (i.e. BER of less than 8%) for the extracted watermark across all benchmarks which is well below the tolerable threshold determined in Section 4.4.2. Table 4.4 also reports the effect of adversarial fine-tuning on the performance of each benchmark.

Table 4.4: Robustness of GradSigns against adversarial fine-tuning.

	CIFAR10	YTF	SVHN
BER (# bit error/total)	5/64	0/64	0/64
Performance difference	-2.58%	+0.23%	-0.22%

Input Noise Injection is an example of input tampering attacks in which the adversary places a random noise on queries to the stolen model to corrupt the gradients approximated through zeroth-order methods, and ultimately prevent watermark verification. For this attack, we consider superimposing a Gaussian noise with zero mean and standard deviation varying from 0.001 to 0.1 with inputs to benchmark models. Note that in all our experiments pixel values of images are scaled to $[0.0, 1.0]$.

Figure 4.3 demonstrates the results of our experiment. As shown, watermarks embedded by GradSigns remain existent and verifiable in presence of input noise injection attack. For all



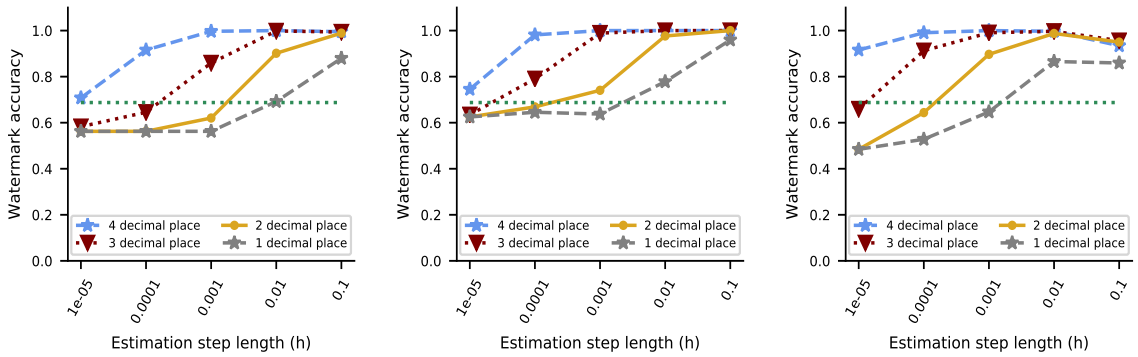
(a) 64-bits YTF (b) 64-bits SVHN (c) 64-bits CIFAR10

Figure 4.3: Resiliency of GradSigns framework against input noise injection attacks. Superimposed noise is drawn from Gaussian distribution $\mathcal{N}(0.0, \sigma)$ with σ varying along the horizontal axis. The green dotted line is the tolerated mismatch threshold.

cases where the superimposed noise is not too large to heavily degrade the model's performance, watermarks embedded by GradSigns can be successfully verified.

Score Rounding is an example of output tampering attacks in which the adversary aims to break the zeroth-order gradient approximation method in GradSigns' verification phase. As mentioned in Section 4.3.2, estimating the gradient of a carrier node c on input sample x through a zeroth-order method involves querying the model with inputs x and $x + he_c$, calculating cross-entropy loss based on the reported prediction probabilities for each query, and finally dividing the observed difference in cross-entropy values by the estimation step length h. The adversary, being aware of our framework, could try to mask the difference observed on the output of the model for these two queries by rounding the reported prediction probabilities, falsify the calculated cross-entropy values, and eventually corrupt the estimated gradients.

Figure 4.4 demonstrates experimental results for score rounding attack on CIFAR10, SVHN and YTF benchmark models. As shown, in occasions that rounding the reported prediction probabilities can obfuscate the correct value of input gradients, and result in unsuccessful watermark



(a) 64-bits YTF (b) 64-bits SVHN (c) 64-bits CIFAR10

Figure 4.4: Resiliency of GradSigns framework against score rounding attack. The green dotted line is the tolerated mismatch threshold.

extraction, increasing estimation step length h to values larger than 0.0001 (which was suggested in Section 4.3.2 for watermark extraction in non-adversarial setting) helps GradSigns extract watermark successfully even for cases where the adversary reports model's prediction probabilities with only one decimal place precision (gray lines in Figure 4.4). Using larger estimation step lengths results in larger differences between reported prediction probabilities for inputs x and $x + he_c$, which would still remain noticeable over the reported rounded probabilities. The horizontal axis of diagrams in Figure 4.4 denotes the value of estimation step length h for the gradient approximation method in GradSigns' verification phase.

Score Perturbation is a more aggressive instance of output tampering attacks in which the adversary deliberately modifies reported prediction probabilities to incur error on the approximated gradients, and prevent watermark verification. While we have considered this category of attacks in our analysis, we believe this category of gradient tampering schemes is less practical for the following reasons: (a) The anti-watermark system cannot tell apart queries from the model vendor and regular clients, therefore, regular users will be impacted by fabricated outputs unintentionally, (b) for models deployed in a critical application such as defense, avionics, and disease diagnosis

in healthcare exact values of prediction scores are required as direct inputs for policy-based decision-making systems.

We evaluate the performance of GradSigns against score perturbation attacks, in which the adversary has two goals: (1) maximally perturbing the probability of top-1 predicted label to corrupt the approximated gradients, (2) minimizing impacts of output perturbations on the model's functionality to preserve its competitive performance and reduce possible harm to non-vendor users. Achieving the first goal equals maximizing the L-infinity ($L\infty$) norm of output modifications. To accomplish the second goal, an adversary should (a) modify as few prediction scores as possible to preserve the fidelity of the reported outputs, and (b) maintain the true order of top-M predicted labels in the fabricated outputs. The latter property is referred to as M-*true label ordering* in the rest of the chapter. In our experiments, M is set to 3.

In the Score Perturbation (SP) attack, the prediction probability of the top-1 label is superimposed with random noise to corrupt cross-entropy values computed for approximating the gradients of carrier nodes. To keep the sum of reported prediction probabilities equal to 1.0, the reported score of other label(s) should also be modified. The adversary, to minimize the impacts of perturbations on the fidelity of reported outputs, only modifies (increases) the prediction probability of one other output label, preferably the label with the least probability.

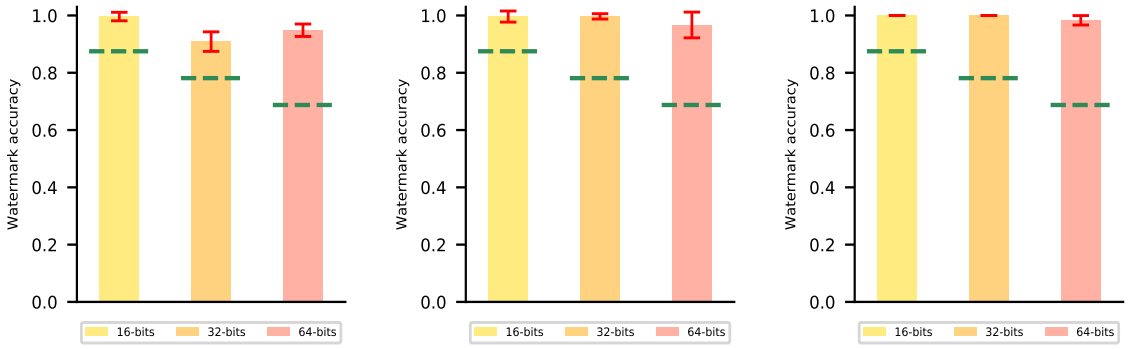
Algorithm 3 summarizes the logistics of SP attack. In lines 2 and 3 of this algorithm, input sample X is fed to the model, and the maximum possible value τ_{sp} which can be reduced from the probability of top-1 label is computed. In our notation, P_{top_j} indicates the j_{th} highest probability in P. Value of τ_{sp} is upper bounded by (P_{top_M}-$P_{top_{|P|}}$) and (P_{top_1}-P_{top_2}) to assure that the M-true label ordering property is held true on the fabricated outputs. In line 4, magnitude (L-infinity norm) of perturbations is randomly decided from a uniform distribution on $(0, \tau_{sp}]$ and

Algorithm 3 Score perturbation attack

1: **Input:** input sample X, model $\bar{H}$, true label ordering M
2: P = $\bar{H}(X)$
3: $\tau_{sp} = \min\{(P_{top_1} - P_{top_2}) - \epsilon, (P_{top_M} - P_{top_{|P|}}) - \epsilon\}$
4: $\phi \leftarrow uniform(0, \tau_{sp})$
5: $P_{top_{|P|}} = P_{top_{|P|}} + \phi$
6: $P_{top_1} = P_{top_1} - \phi$
7: **Output:** P

in lines 5-6, prediction scores of the most and least likely labels (P_{top_1} and $P_{top_{|P|}}$) are modified correspondingly.

Figure 4.5 demonstrates the results of the score perturbation attack for all three datasets. As shown, this attack fails to prevent the model vendor from verifying the embedded watermark as the accuracy of the extracted watermarks are always above the accepted threshold derived in Section 4.4.2. In this experiment, for each benchmark, 50 verification attempts in presence of the SP attack are made, and the mean and standard deviation of the accuracy of extracted watermark over these trials are reported in diagrams in Figure 4.5. The estimation step length is set to 0.1 throughout this experiment for all benchmarks, and ϵ is a small constant set to 0.00001.



(a) CIFAR10 benchmarks (b) SVHN benchmarks (c) YTF benchmarks

Figure 4.5: Accuracy of the extracted watermark in presence of score perturbation attack. The green dashed line is the tolerated mismatch threshold for each benchmark.

4.4.4 Credibility

Credibility requires that (a) finding a ghost (accidental) watermark and (b) forging a secondary watermark into a watermarked model should be impossible, meaning that a watermark exists in the model, *if and only if* it has been deliberately embedded by the model vendor.

If the adversary is able to find a ghost watermark in the stolen model, they can falsely claim that the model belongs to them. To protect GradSigns against *ghosting attacks*, we require the owner to bind their identity to the embedded watermark information. Similar to [106], we suggest that the model owner, instead of using an RNG, generate the watermark b and embedding key K by hashing a meaningful message containing their identity so that even if the adversary is able to find a tuple of $(\hat{b}\ ,\ \hat{K})$ for which the Equation 4.2 holds true, the probability that $\hat{b}$ and $\hat{K}$ are also hash of a meaningful message containing adversary's identity would be close to zero.

Assuming that the adversary is aware of our framework, they might try to forge their own watermark, a *counterfeit watermark*, into the vendor's model, and claim its ownership. To perform the *forging attack*, they have to follow the steps 1-4 discussed in Section 4.3.1, and then re-train the stolen model using their training dataset to minimize GradSigns' cost function (Equation 4.1). In this scenario, we seek to answer the following questions, (1) Is it possible to embed a watermark into a pre-trained model? if the answer to the first question is positive; (2) how big of a dataset does an adversary require to forge their own watermark into the model?

Table 4.5 shows the results of a forging attack in which the adversary tries to embed a counterfeit 32-bit signature into the vendor's 32-bit marked SVHN benchmark. We consider forging a counterfeit watermark using sub-datasets of varying sizes to answer the second question. To embed the watermark, the benchmark model is re-trained for another 80 epochs, the same

Table 4.5: BESR of counterfeit watermark and performance overhead of forging attack.

	Available samples per classification label				
	256	512	1024	2048	**4096**
BESR	0.57	0.60	0.62	0.68	**1.0**
Performance overhead	-1.57%	-1.15%	-4.43%	-3.37%	**-1.09%**

number of epochs required to train the model (from scratch) and embed the original watermark. For each sub-dataset, we conduct the forging attack 3 times with varying λs with values smaller, equal, and greater than what was used for embedding the original watermark, and report the best BESR among these trials on Table 4.5. As shown, the adversary is not able to successfully forge their own watermark unless they have access to a dataset with comparable size to the vendor's original dataset, which is considered out-of-scope in our threat model. As we mentioned before, an adversary with access to a such large dataset ($4096 \times 10 \approx 40k$ samples) has no incentive to hijack the model in the first place as they are capable of training the model from scratch. Even if we assume that the adversary has access to such a dataset, the performance of the model after the forging attempt will be below the stolen model, which points out to inefficiency of forging attacks.

4.4.5 Efficiency

The efficiency metric requires the costs involved in both embedding and detecting the watermark to be affordable.

Our experiments show that the training overhead from using GradSigns' training cost function was only slightly higher than that of the cross-entropy cost function, thus, embedding of a watermark would not affect the time-to-market and design costs of the protected DNNs.

With regards to embedding costs, this requirement gains importance in scenarios where the

stolen model is monetized on an MLaaS [108] platform where customers are charged per-query basis. For the cases where the stolen model is either deployed on MLaaS platforms with run-time based pricing policy, or is commercialized as a learnware, the efficiency metric is not as important since the model vendor is able to make unlimited queries upon purchasing the license. In our method, as mentioned in Section 4.3.2, extracting a watermark involves estimating the expected gradient of the cost function with respect to the carrier nodes, which requires $s \times (|C| + 1)$ queries to the model. Here, $|C|$ is the size of the watermark carrier set, and s is the sample size that determines the number of watermark key images over which the expected gradients are estimated. The standard error of the approximated expected gradients is directly affected by the sample size, in that, the larger the sample size, the smaller the error would be [109].

Note that there is a trade-off between the efficiency and accuracy of watermark extraction. On one hand, to successfully extract the watermark, the model owner needs to compute the expected input gradients with high accuracy which requires a large set of watermark key images, as mentioned before. On the other hand, a large sample size translates to higher verification costs for the vendor. Therefore, to balance the efficiency and accuracy of watermark extraction, the model vendor needs to determine the minimum number of watermark key images (minimum sample size) that guarantees a successful watermark extraction. Column 4 of Table 4.6 reports the minimum sample size required for extraction of the watermark from CIFAR-10, SVHN and YTF benchmark models. These numbers have been experimentally proven to be large enough for a successful watermark extraction over 20 trials. Watermark size, watermark carrier set size and the number of extraction queries per watermark bit for different benchmarks are listed on Table 4.6. Number of verification queries can be reduced by using random-direction based gradient estimation [100, 110] instead of the coordinate-wise estimation method as in Equation 4.5.

Table 4.6: Efficiency of extracting watermark with GradSigns.

Dataset	WM Size	Carrier set size	Min. sample size	extraction queries per WM bit
CIFAR-10	16	128	50	< 500
	32	256		
	64	384		
SVHN	16	256	50	< 500
	32	256		
	64	512		
YTF	16	64	20	< 100
	32	128		
	64	128		

4.4.6 Capacity

Capacity requires the watermarking method to be able to embed a large amount of information into the model. As shown in Table 4.1, unlike contemporary black-box methods which can only embed one bit of information, GradSigns is able to embed watermarks with various sizes of 16, 32, and 64 bits into all CIFAR-10, SVHN, and YTF model benchmarks, which shows our method fulfills this requirement.

4.5 Summary

In this chapter, we present GradSigns, a novel watermarking framework for classification DNNs. GradSigns embeds the owner's signature by imposing a statistical bias on the expected gradient of the training cost function with respect to the model's input. We evaluate GradSigns on DNNs trained with three different image datasets, and experimentally show that (1) our method is extremely robust against general and adaptive counter-watermark attacks and (2) it is capable of embedding a large amount of information with negligible impact on the performance of the model, and (3) it is efficient, reliable and credible, thereby providing a strong proof of ownership.

Chapter 5: IP Fingerprinting: Preliminaries and Literature Review

5.1 The Need for IP Fingerprinting

There is no doubt that watermarking is a vital tool for protecting IP rights, but can it alone deter IP piracy and copyright violations? Arguably, the answer is no, as a watermark cannot provide any assistance in identifying the source of infringement to hold the perpetrator responsible for their actions.

Consider a scenario in which an IP vendor discovers that an unauthorized party is in illegal possession of their IP or is redistributing their work over the market. For ease of explanation, we will refer to the IP vendor as *Alice* and to the pirate as *Bob*. Suppose that Alice's IP is protected through watermarking, and that's in fact how she was able to identify that Bob had committed infringement. Alice could use the watermark as evidence in a court of law to press charges against Bob and eventually prosecute him. But would suing Bob completely resolve Alice's concerns regarding this infringement case? Considering the following, you may see that the answer is complicated.

If Bob is one of Alice's customers, she can conclude that Bob is the source of infringement, i.e. the dishonest customer who has violated their contract and is misusing the IP. Therefore, prosecuting Bob would put an end to the infringement case. However, the matter can become complicated if Bob does not have any clear connection to any of Alice's customers. For instance,

Bob could be an innocent party that unknowingly acquired the copyright to a stolen IP. It is also possible that Bob is merely the front for the infringer to monetize the stolen IP; in this instance, Bob's connections to the infringer would be disguised to protect them against legal repercussions. In such cases, litigating Bob would be just a temporary band-aid solution but not the cure, and as long as the source of the infringement is not identified, a similar case could arise for Alice.

But how can Alice trace the dishonest buyer? The embedded watermark will not help her track down the infringer since all distributed copies of the IP are identical and have the same signature. Alice may seek help from law enforcement to uncover the identity of the infringer. However, such investigations may be costly and take a long time to conclude, especially if the infringer is based in a foreign country. Thus, by the time that the infringer is identified, Alice has already lost a significant share of the market to infringement activities, or worse, the IP has lost its competitiveness and is no longer valuable to her or the market.

Without adequate mechanisms for identifying the source of misuse, IP vendors remain vulnerable to IP infringement activities. Therefore, an effective IP protection scheme should not only allow IP vendors to detect cases of misuse and prove their ownership over IPs but also allow the tracking and identification of IP customers. The former can be achieved through the use of watermarking, while the latter can be accomplished through fingerprinting, which will be discussed in this chapter.

5.2 Preliminaries on IP Fingerprinting

A fingerprint is a unique characteristic of an object that makes it easy to distinguish it from other similar objects. The term fingerprinting describes the process of adding fingerprints to an object and keeping track of them or the process of identifying fingerprints that are already intrinsic to the object.

The concept of IP fingerprinting was introduced in the late 1990s as a means to protect IPs from piracy and misuse. In this context, fingerprinting can be defined as the practice of embedding a unique signature, referred to as *fingerprint*, into each realization (copy) of the IP to allow identification of that copy and the associated customer. The fingerprints identify each copy of an IP in the same way a human fingerprint identifies an individual.

IP Fingerprinting and IP watermarking bear many similarities as both describe information hiding techniques to protect IP rights. In fact, the two share many desiderata and challenges, such as maintaining the functionality of IPs, achieving robustness against removal attacks, etc. In certain cases, such similarities have led to the misunderstanding that the two techniques are equal, which is why we feel it is important to clarify their key difference. *IP watermarking and IP fingerprinting serve distinctly different purposes*. Watermarking is applied for identification and proof of ownership over IPs while fingerprinting is intended to identify the buyer of each copy of the IP. Consequently, in watermarking, the embedded information refers to the IP vendor, whereas in fingerprinting, the payload is to identify the IP customer. Consequently, different copies of an IP share the same watermark, but each copy has a unique fingerprint.

The uniqueness of fingerprints enables the trace of each copy of the IP, including those illegally resold ones. However, fulfilling this requirement presents several challenges for fingerprinting

that do not apply to watermarking: first, fingerprinting requires a large solution space to incorporate a unique fingerprint for each sold copy of the IP. With the number of IP buyers increasing, scalability can become an important challenge for fingerprinting. Second, IP providers most often cannot afford to embed each fingerprint by repeating the entire design process. Designing a large number of different fingerprinted IPs from scratch would have excessive time and cost overhead and thus is not practical for IP vendors. Hence, the challenge is to develop fingerprinting schemes that can provide a large number of distinct realizations of the protected IP with low incremental embedding costs.

5.3 Requirements for an Effective IP Fingerprinting System

A fingerprint, as a means to identify the IP customer, shares very similar desiderata with watermarking. However, there are four requirements that are absolutely essential for IP fingerprinting to be effective:

- **Fidelity (or Loyalty)**: A fingerprinting scheme should have no or negligible impact on the functionality and performance of the protected IP.
- **Scalability**: A fingerprinting scheme should be able to accommodate unique fingerprints for a large number of customers. Typically, this is achieved through multi-bit fingerprints. With a p-bit fingerprint, 2^p different copies of the IP can be identified. Therefore, the longer the fingerprint, the more customers can be supported.
- **Robustness (or Security)**: similar to watermarks, a fingerprint should be robust against removal attacks. Furthermore, the fingerprints should be embedded in such a way that it would be

difficult for a group of customers to collude and create a new copy with an invalid fingerprint from the existing ones.

- **Efficiency**: A fingerprinting scheme should incur low embedding costs on the IP vendor. A large quantity of distinct fingerprinted IPs needs to be generated for different buyers. Hence, the incremental embedding cost (mainly the time and effort) for each instance must be kept reasonably low.

5.4 Current Fingerprinting Schemes for Semiconductor IPs

In Chapters 6 and 7, we will present two fingerprinting techniques for semiconductor design therefore it is important that we review the literature for similar proposals.

Fingerprinting for semiconductor IPs was first introduced in a study by Lach et al. [111]. The authors proposed using FPGA design partitioning and tiling techniques to embed distinct fingerprints. With this approach, the design is divided into several parts, and multiple independent solutions are generated for each part. A fingerprinted IP is then created by choosing the combination of these solutions according to the IP buyer's fingerprint. However, this technique is impractical since it's only applicable to specific IPs with geometric structures. Another fingerprinting technique for FPGAs was proposed by Nie et al. [112] where fingerprints were directly written into unused look-up tables (LUTs) of the bit stream file.

Caldwell et al. presented a generic methodology to embed fingerprints in the solutions to NP-hard optimization problems encountered in IC design [113]. In the proposed approach, NP-hard problems (such as partitioning, Boolean satisfiability (SAT), graph coloring, and standard-cell placement) are first solved to create a "seed" solution. Then, for each IP buyer, a set

of additional constraints are introduced to create a smaller fingerprinted optimization instance. Lastly, the seed solution is incrementally optimized to solve the fingerprinted problem, resulting in a different, but functionally identical fingerprinted IP. Despite the low cost of the small fingerprinted optimization instance, if there are a large number of requests for uniquely fingerprinted copies, there will be a large overhead for IP vendors. Moreover, the proposed approach can not guarantee different solutions.

Qu et al. [114] proposed a generic fingerprinting methodology that introduced additional constraints on the problem formulation of optimization instances in IC design to guarantee a large number of high-quality solutions. While the approach was illustrated on the graph coloring problem, it is applicable to an arbitrary optimization problem encountered in IC design. Roy and Sengupta [115] proposed embedding fingerprints in the high-level synthesis process as constraints on scheduling and register allocation. Imposing signature-based constraints on the circuit partitioning problem has also been explored to encode unique fingerprints [116].

In [117], a satisfiability don't-care condition-based circuit fingerprinting technique is developed to create fingerprints by using MUXs to replace certain library cells. In [118], the same authors proposed to utilize observability don't-care conditions and add extra wires without changing the design's functionality. In both methods, the design will be modified such that fingerprints, in the form of different layouts such as library cells and wires, can be generated at the post-silicon stage. While these methods are practical, they incur large design overhead in circuit area and delay.

The fingerprinting methods discussed so far are classified as static approaches since they require reverse-engineering the design to check whether the constraints generated by the fingerprint are met. On the other hand, dynamic fingerprinting techniques allow the fingerprint to be detected

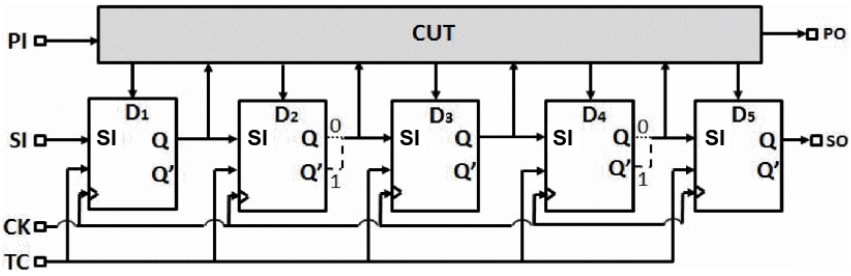


Figure 5.1: Scan chain based fingerprinting [7]. A 5-bit scan chain with the second and fourth connections (from left) chosen as the fingerprinting locations.

by observing the output of the design given a particular input vector. Similar to watermarking, dynamic fingerprinting methods are preferred over static ones.

The first dynamic fingerprinting technique was proposed by Hong et al [119]. In the proposed work, the IP vendor's watermark and IP buyer's fingerprint are fused into a single signature through a blind signature protocol. Then the signature is used to re-code the state variables of test machine FSM in sequential circuits.

In [7], a dynamic scan chain-based fingerprinting scheme with easy detection and low overhead was presented. The proposed method embeds fingerprint bits into the design by adapting different connection styles between adjacent scan cells (as shown in Figure 5.1). Specifically, the proposed scheme chooses the $Q \rightarrow SI$ connection to embed a '0' and the $Q' \rightarrow SI$ connection as a '1' fingerprint bit. It was considered to cause no performance overhead since it made no change to the design except the scan chain. However, because the testing infrastructure is available to the public, this method may be vulnerable.

Chapter 6: A Novel Polymorphic Gate Based Fingerprinting Technique for Integrated Circuits

6.1 Introduction

In this chapter, we propose a fingerprinting scheme for silicon IP blocks using signal-controlled polymorphic gates. Signal-controlled (or input-controlled) polymorphic gates are circuits whose functionality varies based on the value of a control signal. The proposed fingerprinting scheme targets SDC (Satisfiability Don't Care) conditions that usually appear in non-trivial circuits and replaces the standard library cells holding the SDC conditions with polymorphic gates. The modified circuit delivers correct functionality and the configurations of the polymorphic gates constitute the circuit fingerprint. We also introduce additional replacements to those gate locations without SDC conditions. The control inputs of these polymorphic gates serve as dummy fingerprints. Any malicious attempt to modify the dummy fingerprints will bring functional changes and make the fingerprinted circuits function incorrectly.

Our works and contributions are specified as follows:

- Using the evolutionary gate design approach described in Section 3.3, we construct five novel signal-controlled polymorphic gates.
- We propose a polymorphic logic-based justifiability checking method to determine the SDC

conditions and potential locations where standard cells can be replaced by the evolved polymorphic gates to embed fingerprints.

- We evaluate the proposed fingerprinting scheme based on the requirements discussed in Section 5.3 using ISCAS 85 and MCNC benchmark circuits and 0.13um SMIC technology. Experimental results show that the proposed fingerprinting technique is robust, offers high fidelity, and can be used to identify a large number of IP buyers.

The rest of the chapter is organized as follows: Section 6.2 briefly reviews the concepts of polymorphic gates and their design approach. Section 6.3 presents our fingerprinting scheme. Section 6.4, evaluates the proposed scheme based on metrics and properties mentioned for fingerprinting in Section 5.3, and Section 6.5 summarizes the chapter.

The content of this chapter is based on our publication listed as the reference [120].

6.2 Designing Signal-Controlled Polymorphic Gates

Polymorphic gates, introduced by Stoica [78], are novel reconfigurable devices whose functionality varies with changes in their execution environment (such as temperature), supply voltage, or an external control signal. Different from the conventional reconfigurable circuits, which are implemented by multiplexing between multiple stand-alone conventional subsystems, polymorphic circuits need no reconfiguration switch, as the multiple-functionality is inherently embedded within them.

Designing polymorphic gates is extremely challenging, as they exhibit irregular structures which do not follow conventional gate design rules. Consequently, polymorphic gates are typically designed using heuristic-based approaches such as evolutionary algorithms. These algorithms

start by randomly combining transistors, evaluating and ranking them, and combining different partial solutions to eventually create a circuit that completes the desired functionalities.

In Chapter 3, we proposed an automated framework that used the genetic algorithm, a well-known form of evolutionary algorithm, to construct polymorphic gates with any multi-functionality desired. Previously (in Section 3.3), we used this framework to find temperature-controlled polymorphic gates suitable for the watermarking scheme presented in Section 3.4. In this section, we apply the same framework to design another type of polymorphic gate, known as *signal-controlled* polymorphic gates, to use for the proposed fingerprinting scheme.

6.2.1 Evolved Gates

We target two-input one-output signal-controlled polymorphic gates. The evolved gates will have two primary inputs and an additional input that determines which functionality they will implement. The 130nm SMIC technology is adopted for all the evolved gates. We were able to evolve 5 new instances of such polymorphic gates. The functionalities of these gates and their size (in terms of the number of transistors) are summarized in Table 6.1. For example, the first row shows a polymorphic gate with 6 transistors which behave as a NOR gate when the control input C=1, and changes to an inverter when C=0.

Table 6.1: Evolved signal-controlled polymorphic gates.

Gate functionalities	Transistor
NOR(C=1) - INV (C=0)	6
NAND(C=1) - INV (C=0)	7
AND(C=1) – BUF (C=0)	9
AND(C=1) – OR (C=0)	9
NAND(C=1) – NOR (C=0)	9

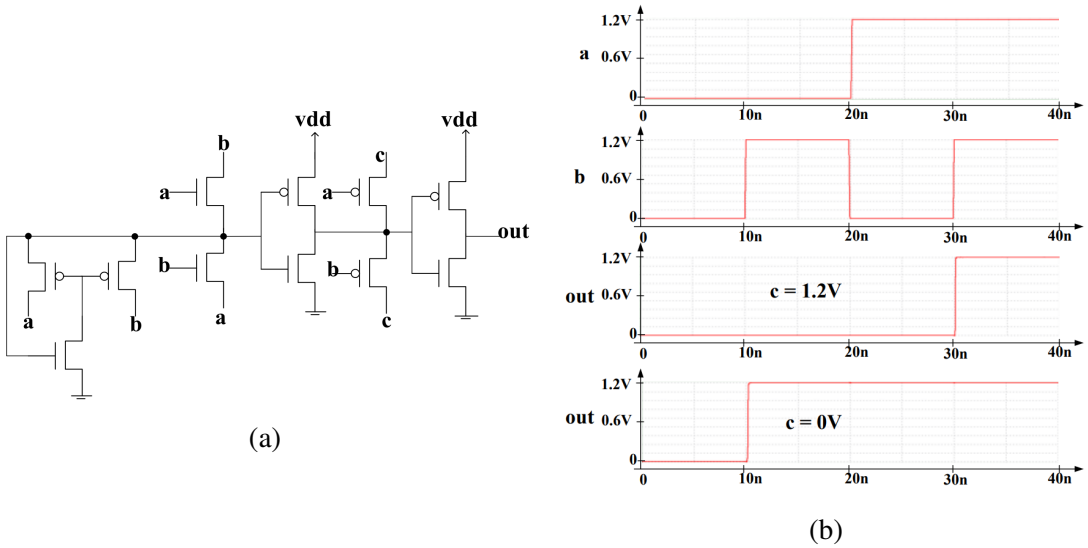


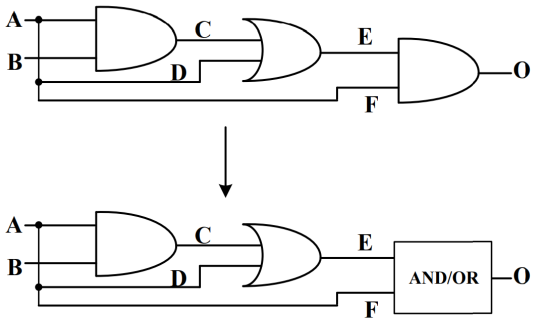
Figure 6.1: Polymorphic AND/OR gate. (a) Topology (b) Input and output wave-forms (from top to bottom: input a, input b, output with c =1.2V, output with c = 0V).

Figure 6.1a presents the schematic of the polymorphic AND/OR gate in the second row from the bottom in Table 6.1. The nine transistors are connected in irregular topology and take unconventional parameters. The function of this gate is shown in Figure 6.1b. When the input c is logic '1', this gate is an AND gate; when c reverses to logic '0', this gate functions as an OR gate.

6.3 Polymorphic Gate Based Fingerprinting Using Satisfiability Don't Care Conditions

6.3.1 Satisfiability Don't Care Conditions

Satisfiability Don't Care conditions describe the logic combinations that will not occur in the internal nets given all the combinations that the primary inputs can take. For example, the AND gate in Figure 6.2a cannot have input patterns (1, 0) or (0, 1) as shown in the truth table in



(a)

A	B	C	D	E	F	AND	OR
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1

(b)

Figure 6.2: An example of SDC condition based fingerprinting. (a) The original circuit (top) and the one after the AND gate is replaced (bottom) (b) Truth tables of the internal signals and gates AND and OR.

Figure 6.2b.

As shown in Algorithm 4, SDC conditions can be determined by using the justification methodology in VLSI testing, which deduces the logic value from an internal net backward to the primary inputs. If the justification of a logic pattern in a gate fails, there exists an SDC condition for this gate. For example, in Figure 6.2a, for the AND gate to have inputs E=0 and F=1, we have A=F=1, but this makes D=1 and the OR gate will make E=1. Therefore, the input pattern (0,1) fails justification and thus it is an SDC condition.

6.3.2 The Proposed Fingerprinting Scheme

In the proposed scheme, we replace standard cells in the netlist with polymorphic gates whose either modes of operation maintain the correct functionality of the circuit. As shown in

Algorithm 4 Determining gate locations with SDC conditions

```
procedure SDC_checking (gate g, pattern < v1,v2, ... ,vn >)
  for (i=0;i<n;i++) do
    if justify ( g's i_th input, v_i ) == fail then
      return pattern (v1,v2, ... ,vn) is a SDC condition;
    end if
  end for
  return fail;
end procedure

procedure Justify (gate g, justification value v)
  if (v != g's existing output value) then
    backtrace();
  else
    if (g functions as AND && v == 1) then
      for (i=0;i<number of inputs for g;i++) do
        if ( justify ( g's i_th input, 1) == fail ) then
          return fail;
        end if
      end for
    end if
    if ( g functions as AND && v == 0) then
      for (i=0;i<number of inputs for g;i++) do
        if ( justify ( g's i_th input, 0) == success ) then
          return success;
        end if
      end for
    end if
  end if
  ...
end procedure
```

Figure 6.2, by embedding such polymorphic gates, we can generate fingerprinted copies of the original circuits, where the configuration of each polymorphic gate represents one fingerprint bit. Note that since polymorphic gates are technology-dependent, the fingerprinting scheme proposed is only suitable for hard IPs targeting the same process technology as the evolved polymorphic gates.

We derive the following principles when determining the potential locations for gate replacement.

Principle 1: Only the gates which have the same logic as either mode of polymorphic gates can be potential locations for replacements. For example, if we want to replace standard cells with evolved AND/OR gate, we should only target the 2-input AND gates and 2-input OR gates.

Principle 2: The SDC conditions of potential replacement locations which we aim to find should be the differentiating input value of the polymorphic gates.

Differentiating Input Value (DIV) of a polymorphic gate is the input combination for which the polymorphic gate produces different output when the control signal changes. For example, input combinations (1,0) and (0,1) are DIVs of the AND/OR gate in Figure 6.1. The output of this gate reverses from 0 to 1 as the control signal changes from '1' to '0' when the two inputs are set as '1' and '0', respectively.

After these gates with such SDC conditions are replaced by polymorphic gates, the functionality of the original circuit remains unchanged no matter which configuration the embedded polymorphic gates take. We assign a binary value to the control input of each polymorphic gate (e.g. the input c in Figure 6.1) which serves as a one-bit fingerprint. By replacing p standard cells with polymorphic gates, we can obtain 2^p distinct fingerprints. In this scheme, we make this type of replacement with the first three polymorphic gates in Table 6.1. These gates possess one DIV that is (0,1), and thus, we need to find standard library gates in the netlist with one SDC condition

of (1,0). Gates with one SDC condition are more likely to appear than the gates with two SDC conditions if we were to consider inserting NAND/NOR and AND/OR polymorphic gates into the original design.

Besides the gates with SDC conditions leading in, we also make some additional replacements for those gates without SDC conditions. To ensure the consistency of the circuit function after modification, the additionally inserted polymorphic gates should be configured to the same logic as the original cells. We take the function control input of these gates as the 'dummy' fingerprint bits as their values are constant and cannot be used for fingerprinting purposes. Any attempt to modify the dummy fingerprints will cause incorrect functionality of the circuit. Therefore, the robustness of the fingerprinted circuits against attacks can be enhanced, as we will discuss in Section 6.4. In this study, we add dummy fingerprints by replacing two-input logic gates with NAND/NOR and AND/OR gates.

The design flow to embed fingerprints is summarized in Figure 6.3. After the locations for replacement are determined, we connect the function control inputs of polymorphic gates to the power supply or ground to configure the functions. Thus, fingerprint values can be embedded in the circuit at the layout design phase. It is important to note that polymorphic gates can also be configured after the layout design stage since they only require minute local rerouting. Additionally, one can configure the polymorphic gates at the post-silicon stage by blowing fuses, or other post-silicon modification techniques. Therefore, the proposed method would not require the designer to repeat the entire design process for each fingerprinted copy.

After the fingerprinted copies of the original circuit are fabricated, we can detect the embedded fingerprints by opening up the chip and identifying the configuration type of the polymorphic gates at each location to recover the corresponding bits in the fingerprints.

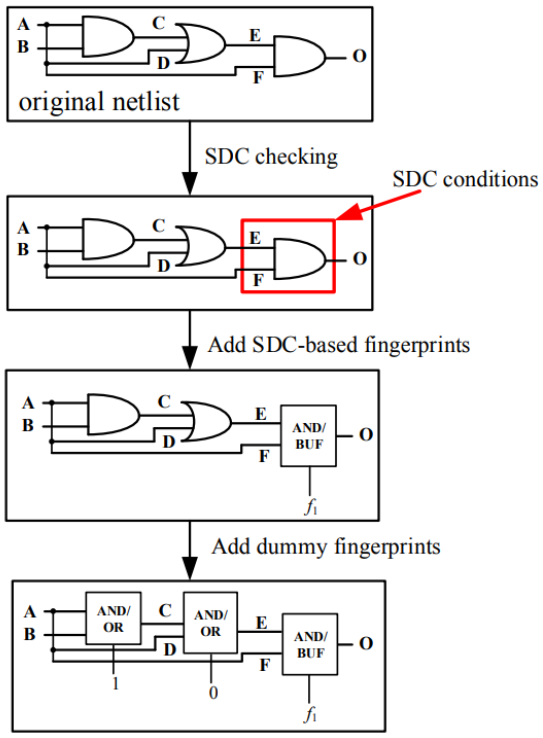


Figure 6.3: Design flow of the proposed scheme.

Table 6.2: Maximum SDC-based and Dummy Fingerprint Size.

Circuit	Gates	SDC-based fingerprint	Dummy fingerprint
C880	290	42	114
C1355	424	69	64
C1908	396	52	106
C3540	943	116	124
C5315	1428	47	548
dalu	1228	52	398
des	3483	70	1712
ex5	609	155	64
i8	1174	208	267
i10	1914	134	509

6.4 Experimental Evaluation and Discussion

In our experiments, we selected several circuits in ISCAS 85 and MCNC benchmarks to embed fingerprints. The circuits are described in Verilog HDL and synthesized using 0.13um SMIC technology and the supply voltage is set as 1.2V. For simplicity, the RTL design is mapped to inverters and two-input/three-input/four-input NAND/NOR/OR/AND gates. A program written in C checks all the SDC conditions and replaces the standard cells with the evolved polymorphic gates automatically. The timing and power of the polymorphic gates are measured using Hspice and their layout area is estimated based on the transistor parameters. With these measurements, we incorporate the polymorphic gates into the SMIC 0.13um synthesis library as standard cells. We collect the area, delay, and power of the original netlists and the fingerprinted netlists using Synopsys Design Compiler.

In this section, we evaluate the performance of our fingerprinting scheme in relation to the properties mentioned in Section 5.3.

6.4.1 Scalability

A fingerprinting scheme should be able to assign a unique fingerprint for a large number of IP buyers. Using the Algorithm 4, we identified all SDC conditions and dummy fingerprint locations for each benchmark circuit. Taking a look at Table 6.2, one can see that in each considered benchmark there are plenty of locations to embed polymorphic gates to carry the fingerprint. This means that our fingerprinting scheme can embed large multi-bit fingerprints and thus is capable of identifying numerous copies.

6.4.2 Fidelity

A fingerprinting scheme must not impact IP's functionality and should have little or no effect on its performance according to the fidelity requirement. In the proposed fingerprinting scheme, polymorphic gates are only inserted in locations that guarantee the circuit's correct functionality independent of the chosen configuration for the embedded polymorphic gates. Therefore, the functionality of each fingerprinted design would be the same as the original design.

The performance of circuits is often measured through factors such as maximum path delay, area, and power consumption. Table 6.3 shows how embedding fingerprints of various sizes impact these metrics for each benchmark circuit. For this experiment, we consider embedding fingerprints of varying sizes (16 and 32 bits) into each benchmark. For both cases, we also add dummy fingerprints of equal width. When we add a 16-bit real fingerprint and a 16-bit dummy fingerprint, the average overhead in delay, area, and power is approximately 2%, 3.5% and 4.3% respectively, which is tolerable; for most benchmarks, performance deterioration is less than 5%. The average overheads rise to 4%, 7%, and 5.1% when we add 16 more bits both in the real fingerprints and dummy fingerprints. Compared to the only other similar approach in [117] where their fingerprints are 32 bits, our average overhead is about only half.

It is evident from the results that the proposed fingerprinting scheme not only maintains the circuit's correct functionality but also has a negligible effect on its performance.

6.4.3 Robustness

When a suspicious IP or circuit appears, the IP vendor needs to recover the fingerprint to trace its source. Attackers who attempt to illegally use the circuit will try to alter the fingerprint

Table 6.3: Area, performance and power overhead of the proposed scheme for fingerprints of varying sizes.

Circuit	16 bit fingerprint			32 bit fingerprint		
	Δ Delay (%)	Δ Area (%)	Δ Power (%)	Δ Delay (%)	Δ Area (%)	Δ Power (%)
C880	1.08	8.11	8.16	1.08	16.31	11.90
C1355	5.72	5.9	7.23	11.74	12.275	8.02
C1908	5.1	6.31	4.13	7.05	13.17	5.09
C3540	0.55	3	2.78	1.11	5.835	4.72
C5315	0	1.645	1.93	6.8	3.355	2.05
dalu	2.38	2.56	2.26	5.71	4.77	2.49
des	4.59	0.77	0.50	4.59	1.455	0.55
ex5	0	3.54	13.05	1.65	6.485	13.26
i8	0	1.845	1.79	0	3.635	2.29
i10	0.74	1.205	0.77	0.74	2.4	0.98
Avg	2.02	3.485	4.26	4.04	6.97	5.14

to prevent it from being recovered. In this section, we will illustrate that our scheme is resilient against such adversaries and provides robust fingerprints in different attack scenarios.

Fingerprint Modification. Given a single copy of the fingerprinted circuit, the attacker can find the fingerprint locations by targeting the polymorphic gates. The attacker can modify the configuration of these polymorphic gates and change the fingerprint to one that has not been distributed by the vendor. However, this attack requires locating the dummy fingerprint bits as changing these bits would influence the function of the circuits. It is not feasible for the attackers to fully reverse engineer the netlist since the fingerprinted circuit is obfuscated with polymorphic gates. Therefore, the attacker cannot differentiate between the dummy and the real SDC-based fingerprint locations, and this attack will become impractical.

Collision Attacks. Collision attack is a powerful attack for all fingerprinting schemes where the attacker has access to multiple fingerprinted copies and can compare these copies to determine the fingerprint locations. More specifically, for our proposed scheme, the attacker may extract the fingerprints and compare their values corresponding to different gate locations. If there

exists a certain location where the corresponding bit value differs in different copies, this location is a real SDC-based fingerprint location and can be modified. A countermeasure to thwart this attack would be encoding the fingerprint before applying it to the original circuit and checking the parity bits in the extracted fingerprint to help reveal any malicious modifications.

SAT-based Attack. SAT-based attacks [121] are powerful adversaries for many logic encryption schemes. As the attacker obtains a gate-level netlist and a functional module of the circuit, he can use the correct input-output pairs to eliminate the wrong configurations of the embedded polymorphic gates for retaining dummy bits, recover the locations of dummy bits, and change them. To gain resilience against these attacks, we can combine schemes like SARLOCK [122], which compares inputs and fingerprint bits to generate a signal that flips the primary outputs. This modification will bring exponential complexity to retrieving the dummy bit locations.

6.4.4 Efficiency

The fingerprint embedding costs in the proposed scheme include the one-time overhead of finding polymorphic gates for the target CMOS technology and determining possible locations to insert the evolved polymorphic gates. After that, the realization of different fingerprints is simply a matter of choosing a unique set of configurations for the embedded polymorphic gates.

Our experiments confirmed that the one-time timing overhead for any of the selected benchmarks was within hours showing that the proposed scheme is extremely efficient.

6.5 Summary

In this chapter, we proposed a novel fingerprinting scheme for ICs to help IP owners track the source of infringements when needed. The proposed scheme utilizes the SDC conditions and makes replacements with the polymorphic gates whose functions are controlled by specific inputs into the gate-level netlist. Experiment results demonstrate that our scheme maintains the functionality of the protected IP, incurs negligible area, delay and power consumption overheads, and can assign a unique signature for a large number of sold copies.

Chapter 7: A Reconfigurable Scan Network Based Fingerprinting Method for System-on-Chips

7.1 Introduction

In the previous chapters, we proposed a fingerprinting method developed for silicon IP blocks to help vendors detect the source of theft when needed. In this chapter, we take on a new challenge and present a fingerprinting technique for system-on-chip IPs, which are commercial designs incorporating more than one IP block.

Fingerprinting an SoC IP is a more challenging task than that of a single IP block. That is because SoCs are typically composed of multiple IP blocks which may have been sourced from external providers. These design blocks are typically shipped as hard IPs that are directly integrated into the physical level, without requiring any modifications. The integration of such hard IP cores into SoCs restricts the design space available to designers for embedding fingerprints.

In this chapter, we propose a fingerprinting scheme that utilizes the standard testing infrastructure in SoCs to create a unique fingerprint for each device. More specifically, we adopt the reconfigurable scan network in SoCs and develop a fingerprinting protocol to configure distinct RSN for each sold copy by utilizing the different connection styles between scan flip flops. The functional test cases for each device will need to be modified consequently to reflect the different RSN

configurations and thus can be used as a fingerprint to identify each device.

Although the different connection styles in the scan chain have been used in the past for IP watermarking [4] and IP fingerprinting [123], they cannot be applied to the SoC IPs because these devices are typically designed by reusing IP cores whose scan chain information is not available to designers. In our approach, we take advantage of the fact that such devices are tested by RSNs and create unique fingerprints at RSN without going into the IP cores. We analyze our approach to show that it will not introduce any design or performance overhead. Meanwhile, our study on the ITC'02 benchmark indicates that the RSN configuration can easily accommodate 10^7 to 10^{186} unique fingerprints.

The rest of the chapter is organized as follows. In Section 7.2, we provide the background knowledge on reconfigurable scan networks. In Section 7.3, we elaborate our proposed RSN based fingerprinting approach. Section 7.4 reports the experimental results with a focus on the potential of the proposed method and its design overhead. Finally, we summarize our contributions in Section 7.5.

The content of this chapter is based on our publication listed as the reference [124].

7.2 Preliminaries on Reconfigurable Scan Network

Scan chains are extensively used to reduce the complexity of functional testing for ICs. They eliminate the need for sequential test pattern generation by making internal memory elements directly controllable and observable. However, in the traditional design of scan chains, where all scan registers are chained into a single scan chain, the time overhead of accessing each module's scan register can be too high. To reduce this overhead, reconfigurable scan networks

are introduced, which enable dynamic reconfiguration of scan networks and allow cost-efficient access to on-chip instrumentation.

In the following, we review the definition of reconfigurable scan networks presented in [125], which covers the existing RSN standards, IEEE 1149.1-2013 [126] and IEEE P1687 (IJTAG).

An RSN has four data ports namely scan-input, scan-output, reset input, and clock input as well as three control ports, capture, shift, and update which are controlled by a 1149.1- compliant TAP [126]. RSNs are composed of scan segments, multiplexers or other combinational logic blocks. The scan segment consists of scan registers which are accessible through the scan-in and scan-out ports, and an optional shadow register. The state of the shadow register determines the configuration of scan networks. Scan segments provide access to testing structures and enable distributed control over the on-chip instrumentation. Each scan segment should support three modes of operations, namely shift, capture, and update, which are controlled by external control signals.

In the capture mode, the scan registers get overwritten by the data coming from the corresponding instrument (Data-in port). During a shift operation, the data from the scan-in port is shifted through the scan registers to the scan-out port. In the update mode, the data in scan registers is written to the optional shadow register. The scan segment might have another control port called select which determines whether the scan segment can perform capture, shift and update operations.

Scan segments are connected either by buffers or scan multiplexers. The latter selects the path that scan data goes through in the network, and its select signal is referred to as the address in the scan network literature. The internal control signals of scan segments such as select, and

the addresses of scan multiplexers are determined by the output of combinational logic blocks whose inputs are the value of shadow registers of scan segments and the primary data and control inputs of the RSN. A scan path is active if all the scan segments on the path are selected, and the addresses of all on path scan multiplexers are set appropriately. To access a scan segment in the RSN, it needs to be on an active path. A read or write access to a scan segment, as defined by IEEE 1149.1 [126], is a three-step process called a CSU (Capture-Shift-Update) operation: in the capture mode of a CSU, all the scan registers on the active scan path load the test result from their corresponding instrument. Then, this data will be shifted out during the followed shift operation. Note that during shift operation, the new scan data will be shifted in as the data in the scan register is being shifted out. Finally, in the update mode of a CSU, the content of scan registers on the active path gets loaded to the corresponding shadow registers.

7.2.1 Segment Insertion Bit Based Reconfigurable Scan Network

Segment Insertion Bit (SIB) is a hardware component proposed by IEEE P1687 [127] which can be used to reconfigure scan networks by bypassing or including scan chains in scan paths. In scan networks, SIBs are utilized to provide fine-grained configurable access to scan chains of instruments and their corresponding sub-modules. A possible implementation of the SIB is proposed in [8] which is shown in Figure 7.1.

A SIB has a scan-input and a scan-output as well as four control inputs, capture, shift, update and select. It also contains a 1-bit shift register S and a 1-bit shadow register U. Note that the same set of external control signals drives scan segments and SIBs in a scan network. During the shift operation, based on the value of shadow register U and the select signal, the data from

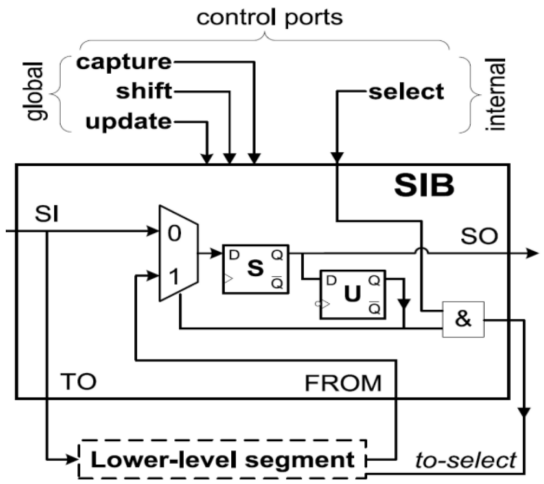


Figure 7.1: Implementation of a segment insertion bit [8].

the scan-in port either gets directed to the lower level scan segment of the SIB (Directing mode) or bypasses the scan segment and directly goes to the scan-out port (Bypassing mode). The value of shadow register U only gets updated from S if both update and select signals are activated. The capture operation is the same as scan segments.

The proposed fingerprinting scheme is based on RSNs. We will elaborate in the following section on how it can be integrated into the SIB-based RSNs, however, it could be applied to other implementations of RSN as well.

7.3 The Proposed Fingerprinting Scheme

Our fingerprinting scheme is built on top of the SIB-based RSNs. It takes advantage of the fact that shift register S and shadow register U in each SIB can be chained by either the Q-D or the Q'-D connection style [4, 128]. In this approach, if the Q-D connection is used to chain S and U registers, the embedded fingerprint bit is '0', and if the Q'-D connection is used, the corresponding fingerprint bit would be '1'. Therefore, for each SIB in the design, one fingerprint

bit can be embedded.

Suppose that the original design only uses Q-D connections for all SIBs in the RSN. Then, the fingerprint of this design would be all 0s. To generate a new fingerprint, the designer has the option of choosing among existing SIBs to modify their S/U connection styles. If k SIBs exist in the design, the designer can create unique digital fingerprints for up to 2^k customers.

As one might notice, when a Q'-D connection is used for the S/U connection of a SIB, the negated value of S will be loaded to U during an update operation, which would make the original test inputs incorrect. Therefore, to ensure that all the instruments can be tested correctly, we need to adjust the test vectors for scan segments whose SIBs have been modified (Q'-D connection is used for their S/U registers). The adjustment only needs to be made to the test input which is shifted in during each update operation. We refer to this test input as a configuration sequence as it determines the scan network topology after its corresponding update operation. To adjust each configuration sequence, the following rules need to be followed for each bit in the sequence.

Rule 1: If the bit corresponds to a SIB whose S/U connection style is Q'-D, the value of this bit should be set to '0' for activating the directing mode and to '1' for enabling the bypassing mode.

Rule 2: If the bit corresponds to a SIB whose S/U connection style is Q-D, the value of this bit should be set to '1' for activating the directing mode and to '0' for enabling the bypassing mode.

These rules make sure that no matter what the style of S/U connection is in each SIB, always the correct value is stored in the shadow register and scan networks can be configured correctly. In the scan network depicted in Figure 7.2, suppose that the original design uses Q-D connections for all three SIBs, i.e. the design carries a fingerprint value of '000'. In this case,

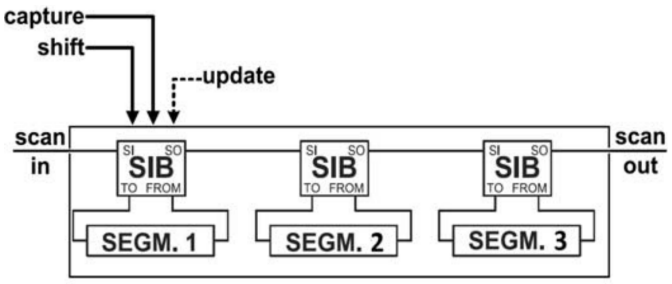


Figure 7.2: An example SIB based RSN for demonstrating test input adjustments.

to access only scan segments 1 and 3, a configuration sequence of '101' should be shifted in before the update operation. As mentioned before, this configuration sequence only works for this specific fingerprint, and if the S/U connection style of any SIB changes, this sequence needs to be modified. For example, if a fingerprint equal to '101' is assigned to the scan network in Figure 7.2, the configuration sequence for accessing scan segments 1 and 3 would be '000'.

To implement the presented fingerprinting method, we propose to replace each original SIB in the design with a slightly different version of SIB called ID-SIB. The only change we made on the original SIB is that the connection style of ID-SIB's S and U registers can be programmed pre-fabrication through local rerouting or at the post-fabrication stage through fuses or circuit editing techniques. In Figure 7.3, if the designer blows fuse F2, the S/U connection will be a Q-D style, and the corresponding fingerprint bit for this ID-SIB would be '0', and if she chooses to blow the other fuse, the connection would be of Q'-D style, and the fingerprint bit would be equal to '1'.

7.4 Experimental Evaluation and Discussion

To evaluate our fingerprinting scheme, we use the SIB-based RSN benchmarks described in [125] which are based on ITC'02 SoC benchmark set [129]. Each ITC'02 benchmark circuit

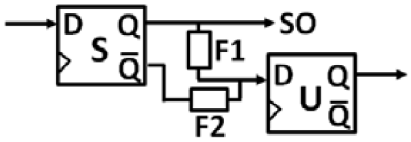


Figure 7.3: Programmable connections of S and U registers in ID-SIB.

is specified by the modules in the SoC and their hierarchical structure, and modules are described by the numbers of their input, output, bidirectional terminals, scan chains and their lengths, test sets, and the (x, y) coordinate of their center on the SOC layout.

In the SIB-based scan network benchmarks, two scan registers are designated for the input and output pins of each module. In this design, doorway SIBs include or exclude lower-level submodules, and instrument SIBs connect or bypass scan segments, input and output scan registers of each module from the active scan path as described in [127]. In Table 7.1, the details of the ITC'02 SoC benchmarks and their corresponding SIB-based RSN designs are listed.

In this section, we evaluate the performance of our fingerprinting scheme in relation to the properties mentioned in Section 5.3.

7.4.1 Scalability

As described in Section 7.3, to embed the fingerprint bits, each SIB in the scan network needs to be replaced with an ID-SIB. Therefore, the number of potential fingerprint bits is equal to the number of SIBs in its scan network, which is given in Table 7.1. As one can see in Table 7.1, with the exception of q127110, all the other benchmark circuits can potentially embed a good number of fingerprint bits with a minimum of 40 bits (A586710) and a maximum of 621 (P93791) fingerprint bits, which correspond to 1.09×10^{12} and 8.70×10^{186} unique fingerprints, respectively. Even in the minimum case, 1.09×10^{12} is a couple of orders of magnitude higher

Table 7.1: Characteristics of the ITC'02 Benchmarks and their corresponding SIB based Scan Networks.

Designs	Characteristics of the ITC'02 Benchmarks				Number of SIBs	Number of unique device fingerprints
	Modules	Levels	Scan Segments	Register Bits		
u226	10	2	40	1,416	50	1.13E+15
d281	9	2	50	3,813	59	5.76E+17
d695	11	2	157	8,229	168	3.74E+50
h953	9	2	46	5,586	55	3.60E+16
g1023	15	2	65	5,306	80	1.20E+24
f2126	5	2	36	15,789	41	2.19E+12
q127110	5	2	21	26,158	25	3.35E+07
p228110	29	3	254	29,828	283	1.55E+85
p34392	20	3	103	23,119	123	1.06E+37
P93791	33	3	588	97,984	621	8.70E+186
T512505	31	2	128	76,846	160	1.46E+48
A586710	8	3	32	41,635	40	1.09E+12

than what's required in most real-life applications.

7.4.2 Fidelity

The proposed fingerprinting approach has negligible performance overhead as embedding fingerprint bits only requires minute modifications in the testing infrastructure of the SoC design which would not affect the performance and functionality of the design during its normal mode of operation. Moreover, the overhead incurred on testing instruments is also negligible since no extra hardware is integrated into the design, and all the changes are done through local rerouting (or circuit editing). For different RSN configurations, the testing vector can be justified, so there will be no change in test coverage.

7.4.3 Robustness

To analyze the robustness of the proposed fingerprinting scheme, we considered both fingerprint modification and removal attacks. For the proposed scheme, the removal attack can

be perceived as an instance of the modification attack, for removing the fingerprint, in other words, changing all the Q'-D connections in SIBs back to Q-D connections can be viewed as a modification attack targeting the fingerprint of all 0s. Therefore, in this section, we only focus on fingerprint modification attacks.

In modification attacks, the adversary's goal is to change the fingerprint of the design. One possible motivation for an adversary to mount these types of attacks is to resell the SoC design to blacklisted customers for higher prices and avoid getting detected by the original vendor. If the digital fingerprints of illegally distributed copies are not modified, the identity of the rogue customer can be easily tracked by the fingerprints.

An adversary can mount modification attacks, only if he is capable of depackaging, reverse engineering the chip, and changing the connections of S and U registers in ID-SIBs using advanced circuit editing techniques. While we believe these assumptions about the capabilities of adversaries are not realistic, especially in the case of very large-scale ICs, we suggest choosing fingerprint bits by the data integrity technique proposed in [7] to eliminate the possibility of such powerful attacks. Based on this technique, embedding fingerprint bits for an IC is a 4 step process: (1): choose N fingerprint bit locations, and replace the corresponding SIBs with ID-SIBs, (2): choose random values for m fingerprint bits with m $<$ N, (3): use this m-bit fingerprint and an IC-specific key (K_{IC}) as the input to a one-way hash function to generate (N-m) bits, (4): use the final N bits as the fingerprint bits to guide the selection of S/U connection styles at the selected fingerprint locations. In this technique, the location of the m-bit fingerprint and the value of the K_{IC} should be kept private to the IC vendor.

The proposed data integrity technique makes it difficult for the attacker to forge a chip fingerprint, since a successful forgery requires knowing the value of K_{IC} and the exact location

of the m-bit fingerprint. The attacker who has access to enough fingerprinted devices may be able to compare these copies to determine the fingerprint locations but without knowledge of the K_{IC} it will be challenging for them to make the correct changes that can maintain the property between fingerprint bits.

7.4.4 Efficiency

In the proposed fingerprinting scheme, embedding costs are low because the fingerprint bit locations can be selected before fabrication, and digital fingerprints can be assigned both before and after fabrication.

7.5 Summary

In this chapter, we propose a novel fingerprinting approach that compared to other existing schemes, is more practical, has lower design overhead, and provides a non-destructive verification method. This method takes advantage of the difference in connection styles in the scan chain to create unique device fingerprints. The testing vectors will be justified accordingly to maintain the test coverage, which becomes one way for the authentication of the fingerprint. It can be conveniently implemented on SoC IPs to utilize their testing infrastructure compliant with IEEE 1149.1-2013 and IEEE P1687 (IJTAG). Experimental results indicate that on standard benchmark circuits, we can generate unique fingerprints a couple of orders of magnitude higher than what typical applications would need.

Chapter 8: Machine Learning in Real-world Applications: Challenges and Solutions

8.1 Introduction

Machine learning models, particularly deep neural networks, are state-of-the-art in many tasks such as image classification, object detection, natural language processing, speech recognition, etc. Nevertheless, deploying DNNs in many real-world applications remains a concern as these models suffer from unexpected failure modes as well as adversarial vulnerabilities.

Our discussion in this chapter will focus on two major safety and security concerns with the use of DNNs in real-world applications namely (a) DNNs' incorrect yet overconfident prediction for anomalous inputs drawn far away from the distribution of training samples, and (b) their vulnerability to adversarial attacks targeting their integrity. A detailed discussion of these concerns will follow, along with a review of solutions proposed to address them. We conclude the chapter by debating why existing solutions may not be practical for commercial DNNs that are shipped as black-box oracles, and why new alternatives are required.

8.2 Challenges in Applying DNNs to Real-world Applications

8.2.1 Inaccuracy and Overconfidence in Presence of Anomalous Inputs

DNNs are traditionally trained based on the ”closed-world” assumption [130, 131], under which the testing data is assumed to be sampled independently and identically from the same distribution as the training data, known as *in-distribution* (ID). However, when DNNs are deployed in ”real-world” settings, more often than not, anomalous inputs arise that do not belong to the known ID. Faced with such inputs, DNN classifiers tend to behave unpredictably and make inaccurate yet overly confident predictions, which can lead to catastrophic consequences in the case of safety and security-critical applications such as autonomous driving cars, medical examinations, and industrial control systems.

In this thesis, we are concerned with the two common categories of anomalous inputs namely *out-of-distribution* samples and *adversarial examples*, which will be discussed in the following sections.

8.2.1.1 Out-of-Distribution Samples

Out-of-distribution (OOD) samples are natural inputs that come from completely different distributions than the model's training dataset. For example, for DNNs deployed in autonomous vehicles, unusual or unknown road signs, rare objects, or scenarios that were not included in the training dataset [132] are considered OOD. Similarly, for a DNN that is trained to classify images of digits, images of English letters are considered OOD. Besides OOD samples that may arise unintentionally in an open-world environment, attackers can also make use of OOD samples

to evade machine learning algorithms. For example, Sitawarin et al. [133] demonstrated that traffic sign recognition DNNs can be easily fooled by OOD data, and outlined two algorithms for generating such samples.

8.2.1.2 Adversarial Examples

The second category of anomalous inputs is adversarial examples, which are deliberately created by an adversary to fool DNNs. Szegedy et al. [17] were the first to highlight this vulnerability inherent to machine learning models. The authors demonstrated that an adversary can modify any input image by adding carefully crafted perturbations such that the modified image, referred to as an *adversarial example* (AE), is misclassified by the model with high confidence. Two interesting aspects are reported for adversarial examples. First, they appear visually similar to the original samples, in that, the added perturbations are virtually undetectable by humans. Second, it was shown that adversarial examples are transferable, meaning that AEs generated for one model are misclassified by another model (trained for the same learning task) regardless of differences in their architectures or training datasets.

Since the nominal work of Szegedy et al.[17], numerous studies [18, 134, 135, 136, 137, 138] have proposed different methods to generate AEs. These techniques, which are often referred to as *adversarial example attacks* or *evasion attacks*, are broadly classified as white-box and black-box methods based on their underlying assumption about the adversary's capability. White-box AE attacks assume the adversary has full knowledge of the model's training data, architecture, weights, and outputs. They often formulate the construction of AEs as an optimization problem in which, given the original input and the target model, the attacker seeks to find a

perturbation that leads to misclassification of the adversarial input while minimizing the perturbation in terms of an appropriate distance metric. These optimization problems are solved using gradients from the model. In the black-box scenario, the adversary can only observe the model's external behavior, i.e its input and output. Black-box AE attacks may use zeroth-order estimation methods to approximate model gradients to solve the same optimization problems as white-box AE methods [137, 139]. Alternatively, a substitute model similar to the target model may be used to generate AEs, as AEs are transferable [140].

It is important to note that AE attacks are not limited to image classification, in fact, over the years a wide range of learning tasks such as speech recognition [19], malware detection [20], medical imaging [21], object detection [22], etc., have been shown to be susceptible to AE attacks.

Both AEs and OOD samples pose a grave risk to the safety and security of DNNs deployed in real-world applications. It is crucial that model users are provided with a mechanism for detecting and mitigating such anomalous inputs.

8.2.2 Threats from Model Tampering Attacks

In addition to the inputs of DNNs, the models themselves may be subject to adversarial attacks. For example, an adversary may try to tamper with the face verification model in an access control system so that it identifies him as an authorized individual. Often referred to as *model tampering* or *model integrity* attacks, these malicious attempts alter the parameters of models at the deployment stage to accomplish their goal.

Model tampering attacks are a common threat for models deployed on both cloud servers

and local devices. For example, an adversary can exploit potential vulnerabilities in security protocols of cloud infrastructures [141] to gain access to the deployed model and substitute its parameters with malicious ones. Vulnerabilities in network protocols may also be exploited to provide access to models in transit. Attackers may also use remote fault injection techniques such as RowHammer [24] and VoltJockey [25] attacks to tamper with models stored on cloud servers.

It is also possible to tamper with models deployed on edge devices. Owing to rapid advancements in model compression techniques [66, 142], DNNs can be deployed on resource-constrained edge devices. Such devices normally do not pose integrity checking mechanisms, leaving the deployed models susceptible to memory fault injection techniques such as laser beaming [26], electromagnetic radiation [27], and voltage glitching attacks. The adversary can leverage such advanced fault injection techniques to perturb the value of model parameters stored on the memory of these devices and compromise the model's integrity.

Several adversarial attacks [61, 143, 144, 145, 146] have been proposed that modify a minimal set of model parameters such that the tampered model misclassifies certain input samples or implements a backdoor. These methods typically use a heuristic search or an optimization-based approach to identify a set of critical parameters that need to be modified to achieve the adversary's goal. The alarming aspect of such attacks is that the tampered model often exhibits similar performance on normal inputs as the original model, thus concealing any evidence of tampering.

As DNNs are increasingly used in high-stakes applications, adversaries find greater and broader incentives to perpetrate such malicious attacks. Needless to say, deploying DNNs to real-world applications without a proper way to validate their integrity can have severe consequences, specifically in safety and security-critical sectors. It is crucial that model users are provided with

a mechanism to verify whether their model has been tampered with.

8.3 Survey of Acknowledged Solutions

Following our discussion of two major safety and security concerns associated with the application of DNNs in real-world production systems, we will turn to the solutions that have been proposed to address these issues.

8.3.1 Anomaly Detection and Mitigation

In recent years, various countermeasures have been proposed to address concerns regarding adversarial examples and out-of-distribution samples. The proposed solutions include anomaly detection techniques that focus on identifying and rejecting these anomalous inputs as well as mitigation techniques that propose specialized strategies to train robust classifiers. In the following subsections, we review the literature related to such proposals.

8.3.1.1 Countermeasures for Out-of-distribution Samples

Model Agnostic Countermeasures: Several studies have presented OOD detection techniques that only take the inputs into consideration and do not rely on any information from the DNN classifier. We refer to these techniques as *model agnostic* methods. A number of proposals [147, 148, 149] utilize the input reconstruction error as a metric to identify ODD data. These techniques are based on the observation that encoder-decoder frameworks (such as autoencoders and GANs) that are trained on the ID data cannot reconstruct OOD samples comparatively, and often result in large reconstruction errors, which can be harnessed to distinguish between ID and

ODD data.

In another category of model agnostic detectors, the distribution of the ID data is modeled and the likelihood under the estimated density is used as a metric to identify OOD samples. These methods often use deep autoencoders and GANs to project samples into a low-dimensional latent variable space and then apply classic density estimation methods in the latent variable space to detect OOD data [150, 151, 152]. However, training GANs and autoencoders often involve tedious hyperparameter tuning (for loss terms and latent space size), which can be computationally expensive.

Countermeasure for Pre-trained Classifiers: A large number of OOD detection techniques assume that the base classifier is pre-trained and rely on information from the model to detect anomalous data. These methods are very popular because they can be applied to many pre-trained DNNs already deployed in the wild. The seminal work of Hendrycks et al. [153] is an example of these techniques in which the maximum softmax probability of the classifier's prediction is used as an indicator score to detect OOD data. This method is based on the intuition that ID data tend to result in higher softmax probabilities than OOD data.

Liang et al. proposed ODIN [154] which used temperature scaling and a gradient-based input pre-processing step to widen the gap between the softmax probabilities of ID and OOD data, thereby improving the OOD detection method proposed in [153]. However, ODIN requires access to auxiliary OOD data to fine-tune the parameters for temperature scaling and the input pre-processing step, which can be a limitation. Ideally, OOD detectors should be completely agnostic of the distribution of anomalous data in order to generalize well to unseen samples encountered in the wild. Hsu et al. [155] proposed Generalized ODIN in order to overcome this

drawback in ODIN.

ODIN is not the only study that uses information from the gradient space to detect OOD data. In [156], Huang et al. proposed GradNorm, an OOD detection method that uses the vector norm of gradients back-propagated from the KL divergence between the softmax output and uniform probability distribution to identify OOD data. GradNorm is based on the intuition that the magnitude of gradients is often higher for ID data than that for OOD data.

Several studies [157, 158] have proposed using energy scores instead of softmax prediction scores for OOD detection. In particular, Liu et al. [157] presented evidence that energy scores better distinguish ID and OOD data compared to softmax scores and showed how energy scores can be extracted from pre-trained DNN classifiers. Reference [158] proposed JointEnergy, for multi-label OOD detection, which computes an OOD score by aggregating label-wise energy scores from multiple labels.

A number of studies have proposed OOD detection techniques that leverage latent representation of inputs extracted from the base classifier to distinguish ID and OOD data. For instance, Lee et al. [159] proposed Mahalanobis, which obtains class-conditional Gaussian distributions with respect to the feature map of the base classifier and then computes the OOD score based on the Mahalanobis distance of the test sample's feature map to all class conditional distributions. In [160], means and standard deviations within feature maps of intermediate layers are used to differentiate ID and OOD samples. Abdelzad et al. [161] used a one-class classifier trained over the feature representations from an early layer to detect OOD data. Sastry and Oore [162] used Gram Matrix values of the activity patterns in the intermediate layers to identify OOD samples. Zaeemzadeh et al. [163] proposed embedding feature representation of each known class into a 1-dimensional subspace for OOD detection.

Alternative Training Strategies: The last category of OOD countermeasures proposes alternative training methodologies to either improve OOD detection performance or train models with well-calibrated confidence. Several studies have suggested exposing classifiers to OOD data, or "outliers", during model training to help models learn better confidence calibration. A better confidence calibration essentially means that the predicted softmax scores would better reflect the likelihood of a correct prediction. Hendrycks et al. [164] used a large dataset of real-world OOD samples to regularize a softmax classifier to predict a uniform distribution for OOD samples and showed that the resulting classifier was able to generalize well to unseen OOD samples. Synthetic OOD samples generated by GANs are also reported to have a similar effect [165]. Recently, Zhang et al. [166] demonstrated that exposing softmax classifiers to "virtual" outliers, inputs that are generated by mixing ID and OOD data, may further boost their generalizability to unseen OOD samples. DeVries and Taylor [167] proposed including an additional branch in the DNN classifier and training the model with a multi-task loss to output both softmax scores and a confidence estimate. Adding a "rejection" option to the classifier is suggested [168, 169] to avoid making decisions if the classifier is not confident. Mix-up training is also effective in training DNNs with better confidence calibration [170].

8.3.1.2 Countermeasures for Adversarial Examples

Following the discovery of adversarial examples, the research community has put considerable effort into developing solutions for defending against AEs. Here we review some of these solutions.

Model Agnostic Countermeasures: A number of studies have suggested using data compression

techniques to remove adversarial perturbations on the model input. Dziugaite et al. [171] and Guo et al. [172] have reported JPEG compression techniques can reduce the effectiveness of AEs for image classification models. However, Shin and Song [173] could generate AEs that survived JPEG compression. Hendrycks et al. [174] discovered that the coefficients for low-ranked PCA components of AEs exhibit higher variation than that of clean inputs. Following this observation, the authors proposed thresholding the variance of later PCA components to detect AEs. Gong et al. [175] proposed training a standalone binary classifier on clean and adversarial inputs to detect AEs at the inference phase. However, Carlini et al. [176] demonstrated the existence of AEs that could evade detection by either of the methods proposed in [174, 175]. Meng and Chen proposed a defense named MagNet [177], in which a denoiser autoencoder is trained on clean training data and the reconstruction error of the denoiser is used as a score for detecting AEs. The core assumption for MagNet is that the reconstruction error for AEs will be much larger than clean inputs, which can be harnessed to detect them. However, it was shown that MagNet doesn't scale well for high-resolution images [178].

Countermeasures for Pre-trained Classifiers: A large number of defense proposals rely on information from the pre-trained classifier to detect AEs. In [179], Lu et al. speculated that AEs produce different activation patterns in late-stage ReLUs compared to clean inputs, and proposed a defense, called SafetyNet, that quantizes the ReLU activation of the last layer in the base model and appends a binary SVM–RBF classifier to detect AEs. Li and Li [180] implemented a cascade classifier on features collected from the outputs of convolutional layers to detect AEs. The classifier is trained using clean and AEs samples. Carrara et al. [181] proposed a defense where given any input, a KNN similarity search is performed on deep representations of

training samples in order to measure the confidence in the class predicted by the base classifier. In [182], histograms created from the absolute activation values of hidden layers are combined into a vector, which is used to train an SVM classifier that detects AEs. Mahalanobis [159], a defense recently discussed for detecting OODs, can also be used to detect AEs. In [183], Carrara et al. demonstrated that the trajectory of internal representations of the network during the forward pass can be different for adversarial and normal inputs, and constructed an AE detector based on this observation. Zheng et al. [184] proposed the I-defender, in which the Gaussian mixture model (GMM) is used to model the activation of the hidden fully connected layers for each input class. At test time, for any given input sample, the likelihood of its hidden activation is compared to the threshold of the predicted class to determine if the input is adversarial. Aigrain and Detyniecki [185] proposed Introspection-Net, in which logits of normal and AEs from the base classifier are used to train a three-layer regression DNN that estimates the confidence for the predicted label. Ma and Liu [186] demonstrated that AEs usually exploit both the provenance and the activation value distribution attack channels. The former channel characterizes the instability of the set of activated neurons in a layer with respect to small changes to the activation of neurons in the previous layer. The latter indicates that the activation values of a layer for an AE can differ significantly from those for a normal sample. Following these observations, the authors propose NIC, in which a set of one-class classifiers are trained for each layer to work as invariant models. At test time, each input is processed by all the trained classifiers, and the final prediction is made based on their results. Lust and Condurache proposed an AE detection method named GraN [187]. In the proposed method, for any given input, the gradient of training loss with respect to model parameters is computed, and then the layerwise norm of computed gradients is combined to form the input for a logistic regression model that is responsible for detecting AEs.

Xu et al. [188] proposed using feature squeezing techniques to detect AEs. In the proposed method, for any test sample, three queries will be made to the base classifier: one with the original sample and two with the squeezed versions of the sample, created by applying color bit-depth reduction and spatial smoothing techniques. If the difference between the prediction produced for the original and the squeezed inputs exceeds a predefined threshold, the test sample is declared adversarial. A similar approach was proposed in [189] which squeezed input samples using scalar quantization and smoothing spatial filter.

Alternative Training Strategies: The last category of countermeasures develops alternative training methodologies to train models that are inherently more robust to AEs. Several studies [18, 135, 190, 191] have proposed including AEs with ground truth labels in the training dataset to help models improve their robustness. There are also proposals to include existing AE generation methods in the training process so the model can learn to defend against such attacks [135]. Adversarial training is a term that is often used to describe these techniques. Adversarial training has been reported to make models robust to white-box attacks [135]. However, it significantly increases the training complexity of DNNs and does not offer protection against black-box AE attacks [191]. Another line of defense, which is referred to as gradient masking/regularization, attempt to enhance the training procedure so that the trained model has small (close to 0) gradients in order to reduce the sensitivity of the model to perturbations in the input sample. An example is a study by Lyu et al. [192] which proposed penalizing the gradient of the training loss function with respect to the inputs. Shaham et al. [193] suggested minimizing the loss function over AEs that are generated at each parameter update to increase the local stability of DNNs. Input gradient regularization combined with adversarial training has been shown to improve DNN's

robustness to AEs [194]. Papernot et al. [195] proposed defensive distillation in which the knowledge extracted from an existing model, available in the form of softmax class probabilities for training samples, is used to train the original model. Also, the last softmax layer is replaced with a softmax-like function to hide gradient information from the adversaries. However, it was reported that models created by defensive distillation can be broken by more advanced attacks [136].

8.3.2 Model Tamper Detection

The prior art proposes a variety of techniques to validate the integrity of DNNs in order to address the threat of model tampering attacks. Existing techniques can be classified as *white-box* and *black-box* methods.

White-box methods require access to the model's parameters to verify its integrity. Examples are traditional data integrity checking methods such as error correction code (ECC), MD5 [28], and CRC [29] techniques. Although these methods can be used to validate the integrity of DNNs [196, 197], they come with a high overhead cost and thus are not practical.

Recently, Li et al. [198] proposed Radar, a DNN-specific tamper detection technique. Radar groups parameters in each layer for checksum computations. The parameters within each group are masked using a secret key and then an addition checksum is computed to derive a 2-bit signature. At run-time, the computed signatures are checked to detect possible changes to parameters of the model. However, the authors suggested using large group sizes to reduce storage and run time overhead, which can adversely affect Radar's detection performance and result in false negatives. Javaheripi et al. [199] proposed using Pearson hash to extract an 8-bit

signature from the parameters in each layer. The proposed method showed a higher detection rate and lower overhead compared to Radar.

Black-box tamper detection (integrity verification) techniques treat the DNN as a ”black box” during the verification and only rely on the model's external behavior, i.e. input-output responses, to determine if the model has been tampered with. Li et al. proposed DeepDyve [200] a black-box integrity verification method that trains a simpler and smaller model to verify the predictions of the original model. However, the performance of DeepDyve is not investigated against stealthy model tampering attacks. He et al. [201] proposed an optimization-based methodology to transform training samples into “sensitive-samples”, which are inputs that make the model outputs extremely sensitive to its parameters. One can verify whether a DNN has been tampered with by querying the model with sensitive-samples and comparing their reported prediction results. However, sensitive-samples are developed to validate only parameters of the classification layer (final layer) in DNNs and not all the layers. Luo et al. [15] proposed an algorithm to select samples from the training dataset that can offer high verification coverage for model parameters. Additionally, the authors proposed gradient-based test generation techniques which transformed input to minimize their training loss.

8.4 Further Considerations and the Need for New Solutions

While there have been numerous studies addressing the concerns regarding model tampering attacks and anomalous inputs, the vast majority of proposed solutions are not suitable for commercial machine learning models.

As we discussed in Chapter 1, model vendors are increasingly resorting to black-box

commercialization of their products to protect their design against IP infringement attempts. In this setting, purchasing the license for a commercial model only provides buyers with the right to use the model's predictive capabilities, and nothing more. This means that licensed customers cannot access the model's internal details, such as parameters, architecture, intermediate computations, etc., and naturally, customers cannot make any changes to the model. Additionally, in some cases, access to the output of the model may also be restricted to mitigate possible IP infringements through model stealing attacks. Several studies [202, 203, 204] have demonstrated that adversaries can iteratively query a model to extract enough information from the reported class probabilities to train a counterfeit model. As a precaution against such attacks, access to prediction scores of commercial models may be partially or completely restricted by the model vendor.

Having discussed how machine learning models are marketed, let's examine why the majority of countermeasures covered in this chapter, are not suitable for commercial models.

Regarding the existing defenses for anomalous inputs, mitigation techniques that were categorized as "alternative training strategies" must be implemented by the model builder during the design stage. However, since model buyers are not privy to the design process of commercial models, determining whether these mitigation have been implemented correctly may be difficult. For instance, countermeasures that require extending the training dataset to include OOD samples are only effective if a very large and diverse dataset of previously collected OOD samples is available for training. It is unclear how model buyers can verify if a commercial model was trained on such a dataset. Moreover, it is often the case that for each training time defense strategy, a new attack can be found that can break it. For example, it didn't take long for Moosavi-Dezfooli [138] to show that the robustness offered by adversarial training can be circumvented. Therefore, robust training strategies may not provide adequate protection against adversarial threats.

The other major category of existing countermeasure, namely detection techniques that are built upon pre-trained models, are not viable solutions for commercial models either. Implementing these solutions requires access to the model's internal computations (such as the state of the hidden layers and the gradient of model components) as well as the model's logit and prediction scores. It is simply not possible to obtain such information from commercial models, which are often shipped as black-box oracles. This would leave model buyers with no other option than model agnostic solutions, which sadly are either not that effective or incur a high cost.

Regarding the existing tamper detection methods, white-box techniques cannot be deployed for commercial models as they require direct access to model parameters in order to determine if the model has been tampered with. Moreover, they often incur high verification overhead, which makes them not practical for models deployed on resource-constrained devices. Further, most white-box integrity verification techniques, with the exception of a few studies, require complete access to the prediction scores of the model, and thus may not be practical for commercial models.

Without a doubt, there is a need for anomaly and tamper detection techniques suitable for black-box commercial models. Our discussion in the following two chapters will focus on addressing this demand.

Chapter 9: AID: Attesting the Integrity of Deep Neural Networks

9.1 Introduction

This chapter proposes AID, a novel tamper detection technique for classification DNNs. AID generates a set of test cases that can help verify whether a model has been tampered with. The proposed method formulates the test generation procedure as an optimization problem in which test cases are optimized to (1) increase the sensitivity of DNN's top-1 output label to weight modifications and (2) maximize the portion of parameters that can be validated by each test case. We refer to test cases generated by AID as *edge-points* as achieving the first objective requires pushing each test case close to *edge* of the boundary of the decision regions. AID can verify the integrity of DNNs without access to their parameters, which makes it suitable for commercial models shipped as black-box oracles. Model vendors can use AID to create test cases for their products, then ship them to licensed customers to help them detect model tampering.

The rest of this chapter is organized as follows: Section 9.2 describes our novel DNN tamper detection/integrity verification methodology. Section 9.3 presents detailed implementation, models, datasets, and integrity attacks considered in our experiments, and then reviews the performance and evaluations. Section 9.4 summarises the chapter.

The content of this chapter is based on our publication listed as the reference [205].

9.2 The Proposed Tamper Detection Scheme

9.2.1 Overview

As discussed in detail in Chapter 8, an increasing number of attacks exploit the vulnerability of DNNs in the parameter space and adversely influence their performance during the inference stage. Often referred to as *model tampering* or *model integrity* attacks, these malicious attempts often modify a minimal set of model parameters such that the tampered model either misclassifies certain input samples or introduces a backdoor. What is most alarming about these attacks is that they are often stealthy and have a negligible impact on the accuracy of the victim model, thus concealing any evidence of tampering. Without a doubt, model tampering attacks are a serious threat to the safety and security of DNN-based decision-making systems. Therefore, it is essential to verify the integrity of DNNs prior to their use in daily operations.

Nevertheless, this may be challenging for commercial DNNs, which are often shipped as black-box oracles, since users cannot access model parameters to investigate tampering. To address this challenge, we propose a novel integrity verification (tamper detection) scheme for model users with limited black-box access. The idea is for model vendors to generate a small set of test cases for which any compromised model will predict different results than the original model. These test cases and their corresponding outputs from the original intact model can be shared with model users to help them verify if their models are tampered with.

With the proposed approach, verifying the integrity of a DNN classifier $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ would be a tuple of processes $(\mathbb{A}, \mathbb{C})$:

1. The **test generation** process $\mathbb{A}(f) = X$ which takes the DNN f as an input, and generate a

set of input test patterns $X = \{x_1, x_2, .., x_k\}$ for which the following property holds:

$$\forall f' : \mathbb{R}^n \rightarrow \mathbb{R}^m \; s.t. \; f' \neq f \iff \exists x_i \in X \; s.t. \; f(x_i) \neq f'(x_i) \tag{9.1}$$

Remark: The generated set X includes input samples whose corresponding output from any compromised model f' will be different from the outputs generated by the original model f. Input patterns in X alongside their corresponding output from the original model f form model's test set $\mathbb{T} = \{(x_i), f(x_i)\}_{i=1}^{k}$

2. The **verification** process $\mathbb{C}(f_{DUT}, \mathbb{T}) = \{true, false\}$ takes the DNN under test, denoted by f_{DUT}, and its corresponding test set $\mathbb{T}$ as the input, and verifies whether f_{DUT} has been compromised. Formally, verification algorithm $\mathbb{C}$ works as follows:

$$\mathbb{C}(f_{DUT}, \mathbb{T}) = \begin{cases} false & \forall (x_i, y_i) \in \mathbb{T}, \; f_{DUT}(x_i) = y_i \\ true, & \text{otherwise} \end{cases} \tag{9.2}$$

There are a set of minimal requirements that should be addressed to generate a robust set of test cases for DNNs [201]. These requirements are as follows:

- **Effectiveness**: Generated test set should be able to detect any trivial modification to model parameters.
- **Reliability**: A DNN's test set should result in zero false positives. If a DNN has not been subject to any modification, the test set should not detect any breaches.
- **Efficiency**: The number of test cases required for a reliable and effective validation should

Table 9.1: Notations used in Chapter 9.

Notation	Meaning
n	Dimension of inputs to DNN
m	Number of classification labels
L	Number of layers in a DNN
$f_i(x)$	Value of i_{th} output node of DNN f given input sample x
$Z_i(x)$	Logit of i_{th} output node of DNN f given input sample x
h^l	Activation map of l_{th} layer
w^l	Weight of links in l_{th} layer
b^l	Biases of neurons in l_{th} layer
u_i^l	i_{th} neuron in l_{th} layer
$h_i^l(x)$	Activation of i_{th} neuron in l_{th} layer given input sample x
$z_i^l(x)$	Weighted input of i_{th} neuron in l_{th} layer given input sample x
w_{ji}^l	Weight of link connecting u_i^l to u_j^{l+1}

be as small as possible to decrease verification costs for model users. Querying DNNs with samples from original training data may eventually detect integrity breaches, but it would not be cost-efficient.

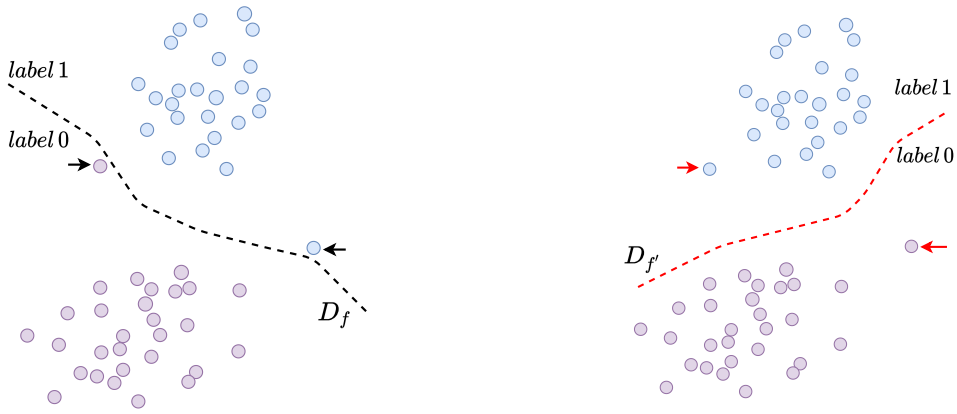
- **Generalizability**: Test generation and verification process should function independently of DNN's architecture, training dataset and application.

9.2.2 Generating Edge-points

In this section, we first discuss challenges in generating robust integrity test cases for DNN classifiers. Then, we describe our test generation methodology, and explain how our approach addresses each challenge. Table 9.1 details the notation used throughout the paper.

We formulate the test generation procedure as an optimization problem in which each test case is transformed to minimize the *AID loss function* $\mathbb{L}_{AID}$ which consists of loss terms specifically included to address the challenges we cover in this section.

Challenge 1: In the AI marketplace, DNNs are often commercialized as black-box oracles



(a) Original boundary D_f (b) Modified boundary $D_{f'}$

Figure 9.1: Breaches to the integrity of a DNN results in shifts to the boundary between classes (D_f). Arrows point to edge-points, which are as shown more likely to be affected by such shifts.

to protect vendors against IP infringements. Model vendors do not disclose the parameters, hyperparameters, or architecture of their model to their customers; instead, they provide them with a prediction API through which customers can send arbitrary inputs and receive outputs. Additionally, access to the output of the model may be restricted due to concerns about model stealing attacks [202, 203, 204]. It has been demonstrated that adversaries can iteratively query a model to extract enough information for training a replicate model. Full access to DNNs' outputs (e.g. confidence scores, top-k labels) gives adversaries a big advantage to mount such attacks. For this reason, vendors may only allow users to view the DNN's final decision output, i.e the top-1 predicted label, to reduce the risks of IP infringements through model stealing attacks. This would further restrict available information about internal computations of models and make it even more challenging to validate the integrity of DNN classifiers.

Considering the aforementioned assumptions, the integrity test cases must be generated in a way that any trivial modification to model parameters would flip DNN's top-1 prediction for them. In what follows, we describe how we achieve this objective.

In the context of classification, model f learns a set of parameters that constitute m decision

regions within the input space, each corresponding to a unique output class. The decision boundary between the m classes of model f, which we denote by D_f, is essentially the set of input samples that reside on the boundary between all classification labels (shown by dashed lines in Figure 9.1 for a binary classifier). Such input samples are equally likely to be classified by f as any output label, as formally defined below:

$$D_f = \{x |\ \forall i \in \{0, 1, .., m-1\}\ \ max(f(x)) = f_i(x)\} \tag{9.3}$$

Modifying parameters of model f results in shifts to its decision regions which manifest themselves as changes to the label assigned to different areas within the input space, especially to regions close to the boundary between classification labels. Inspired by this observation, we require each test case x to lie as close as possible to these regions (as shown in Figure 9.1) so that any breaches to model parameters would be more likely to change the predicted label for test case x. That being the case, L_I loss term is included in $\mathbb{L}_{AID}$ to push each test case close to the boundary.

$$L_I(x, f) \ = \ \sum_{i=0}^{i=m-1} (max(Z(x)) - Z_i(x))^2 \tag{9.4}$$

Here, $Z(x)$ is the logits (non-normalized output) of model f for a given test case x. In our approach, each test case x is transformed to minimize L_I to ensure that model f will output similar confidence scores for all output labels, and thus x resides close to the boundary between all classification labels. As we require each test case to have a top-1 predicted label by the model, we ensure that selected test cases lie close but not exactly on the said boundary.

In order to discuss the other challenge in producing robust integrity test cases for DNNs,

we need to introduce a new concept, namely the *Reactivity Score*. Given an input sample, the gradient of DNN's output with respect to its parameters measures how reactive the model's output is to changes to each parameter. Modifying parameters with large gradients will result in major shifts in the model's output, while changes to parameters with small or zero gradients might not tweak the model's response. Based on this notion, we propose, Reactivity Score (RS), a metric to measure how sensitive DNN's output is to any model parameter.

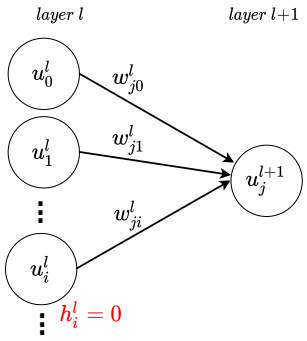
Definition 2: Given an input sample x, *Reactivity Score* of parameter w in model f is defined as follows:

$$RS(w, x) = max \left\{ \left| \frac{\partial f_i(x)}{\partial w} \right| \right\}_{i=0}^{i=m-1} \tag{9.5}$$

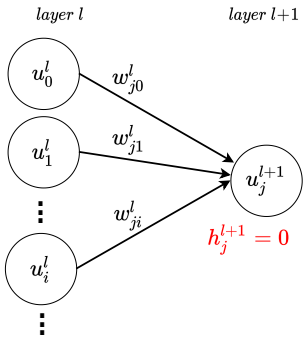
A non-zero reactivity score denotes that small changes to w would cause shifts in the model's response to x, and therefore, testing model f with sample x can validate whether w has been modified. We refer to set of parameters that possess a non-zero reactivity score for test case x as the *Validation Coverage* (VC) of x. Ideally, we want a test case to be able to validate all model parameters. However, as we will demonstrate, that is not possible for DNNs.

Challenge 2: As it was pointed out in [206], provided any input sample, many neurons in DNNs will remain inactive, i.e. output a value equal to zero, and thus, do not participate in output computations. As we will show, given a test case x, incoming and outgoing links connected to inactive neurons hold a zero reactivity score, and therefore are not included in $VC(x)$. The challenge here is how to choose test cases that can activate as many neurons as possible.

Reactivity score of any link such as w_{ji}^{l} in Figure 9.2 can be computed via the chain-rule of backward propagation:



(a) Scenario 1



(b) Scenario 2

Figure 9.2: Computing reactivity scores of w_{ji}^l via chain-rule of backward propagation of DNN.

$$
\begin{aligned}
RS(w_{ji}^l, x) = max\left\{\left|\frac{\partial f_i(x)}{\partial w_{ji}^l}\right|\right\}_{i=0}^{i=m-1} = \\
max\left\{\left|\frac{\partial f_i(x)}{\partial h_j^{l+1}(x)}\right|\right\}_{i=0}^{i=m-1} \times \frac{\partial\, h_j^{l+1}(x)}{\partial z_j^{l+1}(x)} \times \frac{\partial z_j^{l+1}(x)}{\partial w_{ji}^l} = \\
max\left\{\left|\frac{\partial f_i(x)}{\partial h_j^{l+1}(x)}\right|\right\}_{i=0}^{i=m-1} \times \frac{\partial\, h_j^{l+1}(x)}{\partial z_j^{l+1}(x)} \times h_i^l(x)
\end{aligned}
\tag{9.6}
$$

Let's consider two cases: **scenario (1)** w_{ji}^l is the outgoing link to an inactive neuron u_i^l. In this case, h_i^l would be equal to zero, and thus $RS(w_{ji}^l, x) = 0$, **scenario (2)** w_{ji}^l is the incoming link to an inactive neuron u_j^{l+1}, assuming that u_j^{l+1}'s activation function is ReLU or sigmoid function, the common case for most DNNs, gradient of h_j^{l+1} w.r.t z_j^{l+1} would be zero (the 2_{nd} term in Equation 9.6), and thus, $RS(w_{ji}^l, x) = 0$.

In our approach, to increase the portion of model parameters that can be validated by the generated test cases, we require each test case to activate as many neurons as possible to decrease the number of unverifiable parameters, i.e. parameters with zero reactivity score. To this end, we include the term L_{II} in $\mathbb{L}_{AID}$ to penalize inactive neurons for a test case x.

$$
L_{II}(x, f) \;=\; \lambda \sum_{l} \sum_{u_j^l \in l} s\left(-1.0 \times z_j^l(x)\right) \tag{9.7}
$$

Here, λ is a hyper-parameter that determines the contribution of L_{II} to AID loss function, $z_j^l(x)$ is the weighted input of j_{th} neuron at the l_{th} layer given the test case x, and s is:

$$s(x) = \begin{cases} x & 0 < x \\ 0, & \text{otherwise} \end{cases} \tag{9.8}$$

Minimizing L_{II} would maximize the number of neurons that are activated by each test case, and consequently widen their validation coverage.

Algorithm 5 summarizes AID test generation procedure. Line 4 randomly initializes the input x within the allowed range of values denoted by Input_constrained. Line 7 computes the AID loss function. In line 8, the gradient of AID loss function with respect to input is computed to update x with learning rate lr. Next, in line 9, updated x is projected to the allowed input range. Line 12 is included to check whether model f has a single top-1 predicted label for x. As discussed, by including loss term L_I, we push test case x close to but not exactly on decision boundary of m classification labels. We don't accept test cases that reside on the boundary itself as model f can not provide a top-1 predicted label for them. Figure 9.3 shows an edge-point sample for CIFAR10 benchmark and its corresponding output from the DNN.

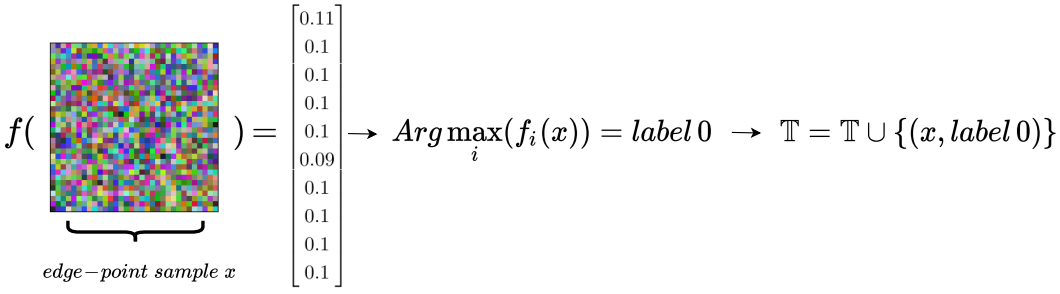


Figure 9.3: An edge-point sample for CIFAR10.

Algorithm 5 AID test generation

Input: Model f, Number of cases to be generated k
$\mathbb{T}$ = {}
while $|\mathbb{T}| < k$ **do**
  x = Initialize(Input_constrained)
  i = 1
  while $i \leq max\_itr$ **do**
    $\mathbb{L}_{AID}(x, f)$ = $L_I(x, f)$ + $L_{II}(x, f)$
    $x = x - lr \times \frac{\partial \mathbb{L}_{AID}}{\partial x}$
    x = Project(x, Input_constrained)
    i = i + 1
  end while
  if $|\{i \mid f_i(x) = \max(f(x))\}| = 1$ **then**
    $\mathbb{T} = \mathbb{T} \cup \{(x, \text{Arg}\max_i(f_i(x)\}$
  end if
end while

9.3 Experimental Evaluation and Discussion

In our experiments, we consider various breaches to the integrity of DNNs. We consider an adversary who seeks to embed a backdoor into the DNN. Three instances of such attacks namely Trojaning [61], BadNet [60], Targeted Trojan Bit (TBT) [144] are considered. Backdoor attacks (also referred to as trojan attacks) co-opt models' learning capability in a way that the model learns both the attacker-chosen sub-task and the primary learning task. A backdoored model behaves as expected for normal inputs. However, when the secret backdoor trigger appears on an input, the model is misdirected to perform the adversary's sub-task(e.g. output a predefined label), regardless of the input's original content.

We also take into account an adversary whose objective is to cause misclassifications for a specific set or class of input samples. Normally, the adversary achieves this goal by poisoning the training dataset with malicious samples. However, recently Liu et al. proposed Gradient

Table 9.2: Details regarding datasets.

Dataset	Images of	# of classes	Features	Pixel range	Model	Accuracy
CIFAR10	Natural objects	10	32*32*3	[0,1]	ResNet20 [104]	90.57%
SVHN	Handwritten digits	10	32*32*3	[0,1]	ResNet20 [104]	96.32%
GTSRB	Traffic signs	43	40*40*3	[0,1]	6 Conv + 2 FC	93.91%

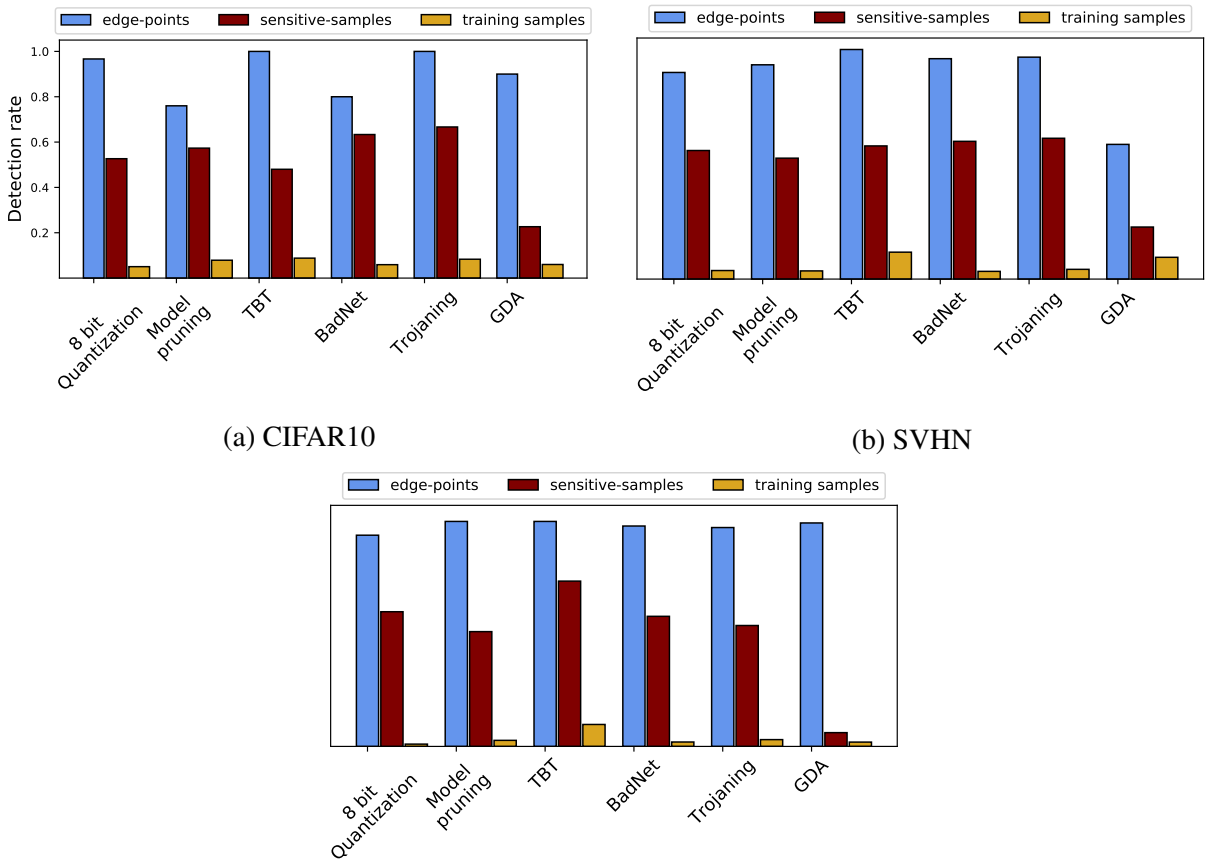
Descent Attack (GDA) [145], a novel technique that enables the adversary to cause targeted misclassification for a set of samples by perturbing a small set of model parameters. Furthermore, we consider integrity breaches through compression techniques such as model pruning [66] and quantization. Despite not being considered as attacks, these methods can show how effective edge-points are for detecting even the slightest changes to model parameters since they maintain the accuracy of the model after the breach.

We experiment with benchmark models trained on CIFAR-10 [101], SVHN [102], and GTSRB [207] datasets. Details regarding these datasets and their corresponding models are provided in Table 9.2. The implementation details and hyperparameters for generating edge-point test cases for each benchmark are reported in Table 9.3. The trade-off variable for each benchmark is selected (over trials) such that both terms in the AID loss function can be minimized simultaneously.

In what follows, we assess the performance of edge-points with respect to the requirements mentioned in Section 9.2.

Table 9.3: Implementation details and hyperparameters of AID.

Dataset	learning rate	λ	max_itr
CIFAR10	0.001	0.25	10000
SVHN	0.001	0.20	10000
GTSRB	0.008 ($\times$0.5 every 2000 epochs)	0.40	10000



(a) CIFAR10

(b) SVHN

(c) GTSRB

Figure 9.4: Comparing the detection rate of edge-points, sensitive-samples, and samples from training dataset against integrity attacks.

9.3.1 Effectiveness

We evaluate how effective edge-points are in detecting various integrity breaches. Moreover, we compare the performance of our method against state-of-the-art sensitive-samples [201]. For this experiment, we generate 150 instances of edge-points and sensitive-samples for each benchmark, and report the detection success rate of generated test cases across various integrity breaches. For a fair evaluation, as we don't require edge-points to resemble samples of training data, we relax this condition for sensitive-samples as well, which would even improve their effectiveness. Results reported in Figure 9.4 denote the ratio of test cases whose top-1 label

predicted by a compromised model varies from the top-1 label predicted by the original model. In addition to sensitive-samples, we report the detection rate of samples from the training data to showcase the effectiveness of our approach.

Following points can be taken from Figure 9.4: (1) Edge-points possess higher detection rates across all considered model integrity breaches compared to the contemporary sensitive-samples. This implies that edge-points are more effective in identifying breaches compared to sensitive-samples, (2) Samples from training data perform poorly in detecting model integrity attacks. That is because model integrity attacks are often stealthy and have a negligible impact on the model's accuracy. (3) GDA attack is exceedingly challenging for sensitive samples to detect and showcases their limitation. This is due to the stealthy nature of GDA which is specifically designed to modify as few parameters as possible.

9.3.2 Efficiency

Efficiency is defined as the number of test cases (model queries) required to guarantee a successful detection. Ideally, we want this number to be as small as possible to decrease verification costs for model users. Querying DNNs with samples from original training data may eventually detect integrity breaches, but it would not be cost-efficient. Table 9.4 reports the minimum number of edge-points required to guarantee a successful detection for any of the considered integrity attacks. For this experiment, we generate 150 edge-points for each benchmark and randomly select test sets of varying sizes $k \in \{2, 3, .., 150\}$. We repeat this experiment 500 times for each k and report the minimum number of test cases that could successfully detect each integrity breach over all trials. As shown in Table 9.4, using less than 5 edge-

Table 9.4: Number of edge-points for a guaranteed successful detection.

Dataset	8-bit quantization	Model pruning	TBT	BadNet	Trojaning	GDA
CIFAR10	3	4	1	3	1	3
SVHN	3	2	1	2	2	4
GTSRB	2	2	2	3	2	1

points we can successfully detect all considered integrity breaches across all three benchmarks. Undoubtedly, our approach incurs negligible verification costs on model users and is practical for real-world scenarios. Light-weight verification costs of AID allows model users to verify their models more frequently which is required for safety and security-critical sectors.

9.3.3 Reliability

Validating the integrity of DNNs with edge-points guarantees a zero false-positive rate, meaning that if the model has not been subject to any modification, edge-points do not detect any breaches. This is because DNNs are deterministic functions whose output relies only on model parameters and the provided input. Therefore, if parameters of a DNN are not modified, the model's output to the provided test cases would be consistent with the expected response.

9.3.4 Generalizability

AID does not make any assumptions about the architecture and application of DNNs and their corresponding training datasets; Therefore, it is in nature applicable to all DNNs targeting classification tasks.

9.4 Summary

In this chapter, we propose AID, a novel methodology to validate the integrity of DNNs. AID generates a set of integrity test cases that can reveal whether a DNN has been compromised over a restricted access to the model's top-1 predicted output label. We evaluate the performance of AID versus a wide range of model integrity attacks and our experiments show that AID is highly effective and efficient in detecting integrity breaches, and offers a major improvement on the state-of-the-art.

Chapter 10: PAD-Lock: Power Side-channel Based Anomaly Detection for Deep Neural Networks

10.1 Introduction

In Chapter 8, we discussed that DNN classifiers can be highly inaccurate yet overconfident when faced with anomalous inputs statistically or adversarially drawn far from the distribution of their training data, known as *in-distribution* (ID). This behavior is both misleading and dangerous, especially for models deployed in safety- and security-critical applications. Therefore, it is crucial to have defensive measures in place to detect anomalous inputs so that fail-safe measures can be triggered and humans can take control of the system.

Additionally, Chapter 8 surveyed state-of-the-art anomaly detection techniques to discover that the majority of existing solutions [159, 161, 162, 163, 179, 180, 181, 182, 183, 184, 186] use information about the model's internal computations (i.e. its intermediate layer representations) to find patterns that could distinguish between anomalous and ID data. While these countermeasures have been proven effective against anomalies such as adversarial examples (AEs) and out-of-distribution (OOD) samples, they can not be implemented for classifiers that are available as black box oracles, which make up the majority of commercial models in the AI marketplace. Additionally, Chapter 8 discussed model-agnostic anomaly detection techniques [147, 148, 149,

150, 151, 174, 175, 177] that only take the inputs into consideration and do not rely on any information from the DNN classifier. While these solutions are applicable to black-box classifiers, they offer subpar performance compared to other model-dependent countermeasures.

Having access to the internal state of classifiers, especially feature maps of their hidden layers, provides valuable insight for detecting anomalous inputs. Such information, however, cannot be obtained from black-box classifiers. To tackle this problem, we propose using side-channel information from the hardware implementation of black-box DNNs to gain knowledge about their inner computations and use that knowledge to protect them against anomalous inputs.

Side channels are inevitable byproducts of running algorithms on microelectronics. Any algorithm implemented on hardware is always accompanied by a number of unwanted physical properties [208]. For example, the device may take a specific amount of time to execute the algorithm, or its power consumption may vary as it works through different stages of the algorithm. If such physical phenomena, known as side-channels, correlate with the sensitive data processed on the device, they can leak confidential information about the algorithm's internal state and possibly compromise its security and confidentiality. Attacks that use side-channel information leakage to extract confidential data are known as side-channel attacks (SCAs). Historically, cryptographic systems have been the primary target of SCAs, and numerous studies have proposed techniques to break both symmetric and asymmetric encryption algorithms [209]. However, recently, ML models, especially DNNs, have been shown to be vulnerable to SCAs. In fact, several studies [210, 211, 212, 213, 214, 215] have shown success in extracting specific properties of DNNs such as their parameters, hyperparameters, and architecture from physical and micro-architectural side-channel emitted during model inference.

The research community often considers side-channel information leakage as a negative

side effect that could compromise the security and confidentiality of algorithms implemented on hardware. Contrary to this perspective, we view side channels as part of the available design space and focus on leveraging them to add useful functionality. Particularly, our research focuses on using the side-channel information of DNNs to improve their safety and security against anomalous inputs. One might say, we want to *"turn the lemons, the side-channel of DNNs, into lemonade, an effective anomaly detection technique"*.

But does the power side-channel of DNN execution have any value for detecting anomalies you may ask? Here is our answer: numerous studies [159, 161, 162, 163, 179, 180, 181, 182, 183, 184, 186] have shown that anomalous and ID data result in contrasting activation patterns on the hidden layers of DNN classifiers. As the power side-channel is known to be highly data-dependent [209], the differences in the activation pattern of intermediate layers will be reflected in the power consumption of the device running the inference. Consequently, we can expect the power side-channel of DNN inference for ID and anomalous data to differ significantly, allowing us to differentiate them at run-time.

To this end, we propose PAD-Lock, a **P**ower side-channel-based **A**nomaly **D**etection framework for black-box DNN classifiers. PAD-Lock uses the power side-channel information during DNN inference operation as an indirect measure of the model's internal state and proposes an auto-encoder-based and a classification-based approach to make use of such rich information for detecting out-of-distribution samples and adversarial examples, respectively.

In this chapter, we present a preliminary analysis illustrating PAD-Lock's potential as an anomaly detection system for DNNs. We implement our method for a multiple layer perceptron (MLP) MNIST [107] classifier deployed on an FPGA-based hardware accelerator. We deploy Time-to-Digital Converter (TDC) power sensors to measure the power consumption of the model

running on the FPGA and use the collected power traces to implement the proposed anomaly detection method. Based on our preliminary analysis, PAD-Lock exhibits promising performance in detecting out-of-distribution and adversarial samples. This work serves as an initial step in studying power side-channel analysis for anomaly detection in DNNs.

The remainder of this chapter is organized as follows: In Section 10.2, we cover preliminaries on FPGA power sensors, particularly TDCs, and review prior studies on power side-channel analysis of DNNs. In Section 10.3, we first review the problem formulation and assumptions behind the proposed methodology. Then, we will cover details of anomaly detection with PAD-Lock. Section 10.4 discusses the experimental setup and implementation details of our proof-of-concept experiments. In Section 10.5, we will go over preliminary results and discuss the future research directions. Finally, Section 10.6 summarizes the chapter.

10.2 Background Knowledge and Related Work

10.2.1 FPGA Power Sensor

The Power Distribution Network (PDN) of FPGAs experiences transient voltage fluctuations when a synthesized logic (the implemented algorithm) becomes active and power is consumed. Given that the propagation delay of combinational logic is inversely proportional to the value of supply voltage, a circuit's delay can be used to indirectly measure the variations in its supply voltage and thereby measure its power consumption.

The propagation delay of FPGAs can be measured using time-to-digital converters (TDCs) [9, 10] or ring oscillators (ROs) [216]. However, TDCs are more commonly used than ROs due to their lower overhead and higher sampling rates. In this work, we will use TDC sensors to collect

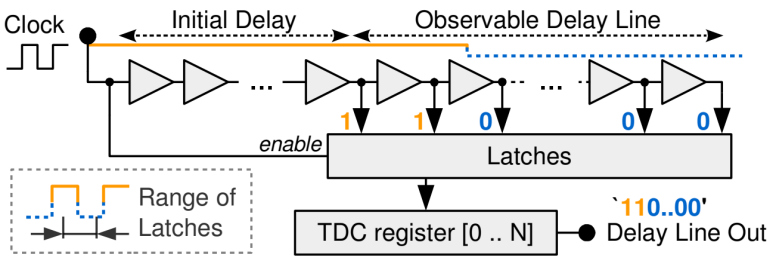


Figure 10.1: Basic design of TDC sensor from [9].

the inference power of DNNs deployed on FPGAs.

As shown in Figure 10.1, a TDC is designed as a long chain of buffers through which a clock signal propagates. Additionally, the output of buffers is tapped with a series of latches that are enabled by the same clock signal. Therefore, the value observed on the latches is an indication of how quickly the clock signal can propagate through the buffers within each cycle. When power is consumed (i.e. the synthesized logic is active), the value of supply voltage drops, which increases the circuit's propagation delay, causing the rising edge of the clock signal to propagate through fewer delay buffers, and thus fewer "1"s will appear on the tapping latches. Alternatively, if the synthesized logic is not active (i.e. is not consuming power), the supply voltage remains high, the propagation delay remains low, and the clock signal can propagate through more delay buffers, thus, more "1"s will be observed on the lathes. Therefore, the Hamming weight of the tapping latches (number of "1"s on the latches) can be used as a proxy for measuring the circuit's propagation delay. In most TDC designs [9, 10], priority encoders are used to calculate Hamming weights of TDC outputs.

Typically, to reduce the area overhead of TDC sensors, only the buffers at the bottom end of the delay line are tapped, since the clock signal always passes through the first few buffers regardless of the circuit delay level. The tapped portion of the delay line is called the *observable*

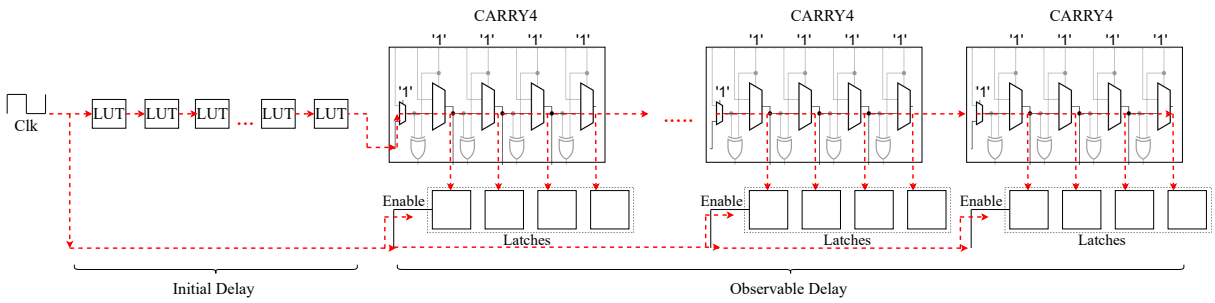


Figure 10.2: A common TDC design using LUT and CARRY4 elements [9, 10]. The red dashed lines demonstrate the path through which the clock signal propagates.

delay line, and the remaining portion is called the *initial delay line* in the literature [9, 10]. For the initial delay line, the buffers are often implemented using LUTs or latch elements, which have a small area overhead and a high propagation delay. Additionally, it is common to replace the buffers in the observable delay line with carry-chain primitives (as shown in Figure 10.2) as the taps on carry-chain blocks are on the order of 5-25 pico seconds apart [217], and allow for higher resolutions per bit.

10.2.2 DNN Power Side-channel Analysis: Current Studies

Recently, a few studies have shown that hardware implementations of DNNs are vulnerable to power SCAs. In [218], the inputs to a convolutional neural network deployed on an FPGA-based accelerator were reconstructed from the power traces measured during the inference operation. In a similar study [217], the inputs to a binarized neural network were reverse-engineered from power traces collected using TDC sensors. In [211], it was assumed that the target model belongs to a set of well-known architectures with pre-trained parameters. Then, an SVM classifier was trained on features extracted from the power traces of well-known architectures (with varying pruning rates) to reverse-engineer the architecture and parameter sparsity ratio of the target model. Zhang et al. [219] proposed a power SCA that could reverse engineer the macro-features

of DNN including the number and type of layers, and hyper-parameters of each layer (e.g number of neurons, filters, stride, activation function, etc.). Tian et al. [220] could extract similar macro-features from power side-channel information of DNNs implemented using versatile tensor accelerators on FPGAs. In [214], a power SCA was proposed to reverse engineer weight parameters of DNNs implemented with systolic arrays, which are matrix multiplication circuits used in hardware platforms such as Google TPU.

To the best of our knowledge, no studies have utilized the power side-channel information of DNN implementations to produce constructive functionalities, thus making our study unique in this regard.

10.3 The Proposed Anomaly Detection Scheme

In this section, we outline our assumptions and formulate the problem we aim to solve. We then describe PAD-Lock, our proposed anomaly detection technique.

10.3.1 Problem Formulation and Assumptions

In this work, we consider the anomaly detection problem in classification settings. Our objective is to identify and reject anomalous data points that come from a distribution different from the classifier's ID. We consider two classes of anomalous input for DNN classifiers. The first category is out-of-distribution samples which are inputs that naturally belong to a distribution other than the model's ID. The second category is adversarial examples, which are generated deliberately by introducing perturbations to ID samples in order to fool DNNs [17]. We'd like to note that a detailed discussion of both categories is provided in Chapter 8.

To ensure that the proposed method is compatible with commercial models we assume that (1) the DNN classifier is only available as a black-box oracle. In other words, the defender, the party who seeks to detect anomalous inputs, has no knowledge of the model's parameters and architecture, and can only send arbitrary inputs to the model and receive the corresponding outputs; (2) the access to the output of the classifier is restricted to its final decision (top-1 predicted label). Additionally, we assume the defender is capable of monitoring the classifier's power consumption while it is performing inferences.

10.3.2 PAD-Lock: Methodology

In PAD-Lock, we view the power side-channel information from the implementation of DNN classifiers as a proxy of their inner processing states and utilize this rich information to distinguish between ID and anomalous data. The proposed framework consists of two components, a **P**ower **S**ide-**C**hannel **C**ollection (PSC2) module, and an **A**nomaly **D**etection (AD) module. The PSC2 module measures the power consumption of DNN during inference operation for any given input sample. For the sake of simplicity, we will refer to the power side-channel of DNN inference for a given input as the input's *inference power*. The power side-channel collection starts when the DNN begins processing the input and ends after the predicted label is returned.

The implementation of the PSC2 module depends on the hardware platform used to run the model. For DNNs running on FPGA-based accelerators, on-chip sensors such as TDCs [9, 10] and ROs [216] can be used to collect power traces in real-time. For models running on different hardware platforms, such as micro-controllers, GPUs, etc, dedicated on-chip power sensors may be used for this purpose. In this study, we use TDC power sensors to measure the inference

power for a DNN deployed on an FPGA-based accelerator. The implementation details of the PSC2 module will be discussed in Section 10.4.

The AD module is responsible for predicting whether a test sample belongs to ID, based on its inference power. This module has an offline (design time) and online (test time) phase. The offline phase starts by collecting the inference power for a set of held-out ID samples, which would allow us to construct a labeled inference power dataset $\mathbb{P} = \{p(x_i), (y_i)\}_{i=1}^{k}$ for the ID data. Here, x_i is an data point from ID with the ground truth label of y_i, and $p(x_i)$ denotes the inference power for x_i. Later in the offline phase of PAD-Lock, the collected power dataset $\mathbb{P}$ is used to develop an auto-encoder-based and a classification-based detector for identifying OOD and AEs samples, respectively.

10.3.2.1 Out-of-Distribution Detection

PAD-Lock implements an auto-encoder-based methodology [221] to detect OOD samples given their inference power. An auto-encoder is an unsupervised deep learning model which is comprised of two components, an encoder, and a decoder. The encoder maps the original data into a low-dimensional latent variable space, and the decoder reconstructs the data from the low-dimensional latent representations. Given a training dataset, autoencoders are trained to reduce the difference (often Euclidean distance) between the reconstructed and the original samples. In this process, the auto-encoder learns an optimal mapping to a low-dimensional space on which any given sample (drawn from the same distribution as the training data) is well represented and can be accurately reconstructed. However, for samples from a different distribution than the training data, the mapping to the latent variable space is not optimized, thus reconstruction with

the auto-encoder results in larger errors. This would allow us to use the reconstruction error of autoencoders as an index to detect anomalies.

For detecting OOD samples with PAD-Lock, at design time, an auto-encoder is trained on $\mathbb{P}$, the dataset of inference power collected for ID data. At run-time, the power inference for each test sample is collected and fed to the auto-encoder, and if the reconstruction error of the sample is higher than a predefined threshold, a flag is raised to trigger further investigations. Note that the detection threshold must be determined by balancing false positives and false negatives. A high detection threshold may lead to many OOD samples going undetected, while a small threshold may incorrectly identify many ID samples as anomalous.

10.3.2.2 Adversarial Example Detection

Numerous studies [18, 134, 135, 136, 137, 138] have proposed techniques for creating AEs, and in all these proposals, the goal is to apply minimal perturbations to samples from ID in order to cause targeted or untargeted misclassification. In a particularly interesting study, Su et al. [222] showed that even modifying one pixel could fool DNNs used for image classification. This indicates that AEs can be visually very similar to ID data, more so than OOD samples, which may be from any random distribution. Naturally, we expect that the distribution of inference power for AEs to be very similar to that of ID data, and thereby, using an auto-encoder based approach for AE detection may not be effective. As a result, we implement a different method for detecting AEs in PAD-Lock.

For detecting AE samples with PAD-Lock, a classification-based approach is proposed where a secondary model, called the *checker model*, is trained on the power dataset $\mathbb{P}$ to classify

the inference power of ID samples into their corresponding classification label. At run-time, the checker model is used to cross-validate the label predicted by the base classifier, and a flag is raised to trigger further investigations if the predictions of the two models are inconsistent.

10.4 Experimental Setup and the Implementation Details

Our experimental setup is based on a NEXYS 4 DDR board, which includes an Artix-7 FPGA with 15850 logic slices, 240 DSP slices, and 4860 Kbits of block random access memory (BRAM).

We implement PAD-Lock using a pre-trained MLP for MNIST [107] classification, which can be found at [223]. The model is available in the Open Neural Network Exchange (ONNX) format, and consists of three hidden layers with 64 neurons and an output layer with 10 neurons, with weights and activations quantized to 2-bit values. It offers a 96.6% accuracy on testing samples of the MNIST dataset. Xilinx's FINN framework [224] is used to deploy the model on our NEXYS 4 DDR FPGA board. FINN is an open-source framework for building DNN inference accelerators on Xilinx FPGAs. It takes in quantized DNNs (in ONNX format) and generates their Verilog implementation for FPGA deployment.

We deploy 6 TDC power sensors to implement PAD-Lock's PSC2 module and measure the model's inference power. Our experiment uses the TDC design shown in Figure 10.2. On that front, each TDC sensor has 16 LUT-based buffers and 16 CARRY4 components chained together to form the initial and observable delay lines, respectively. Four taps are included on each CARRY4 primitive, which enables the sensors to quantize the circuit delay into 64 bins. A 64-to-6 priority encoder was used in the implementation of each sensor to compute its output

(i.e. the Hamming weight of the latches) as a 6-bit binary value. Additionally, we implemented a dedicated first-in-first-out (FIFO) buffer for each sensor to store its output in BRAM units in an orderly manner.

The length of the initial delay line in our sensors was calibrated through experimentation to make sure that (a) the clock signal can reach the observable delay line (i.e. values of latches are not all "0"s) and (b) the clock signal does not reach the final latches to saturate the output of TDC (i.e. values of latches are not all "1"s). This calibration needs to be done at run time as the propagation delay of components used in the sensors varies due to process variations, and thus can not be known at design time. In our experiments, the TDC sensors are spread out across the implementation of the model (as shown in Figure 10.3) in order to monitor supply voltage fluctuations across the PDN more closely. The TDC sensors run at 100MHz and the DNN at 25MHz. This way, the model's power consumption is sampled (measured) four times within each clock cycle. Throughout our experiments, the inference power for each input sample is calculated by averaging the readings from all six sensors across time. Additionally, to reduce the impact of environmental noise on readings from TDC sensors, the inference power for each input sample is collected five times and then averaged.

The FPGA and the PC communicate using the Universal Asynchronous Receiver/Transmitter (UART) protocol to transfer input samples, the model's prediction as well as the TDC sensor readings. The sensors and the UART receiver and transmitter modules are written in Verilog. During the synthesis procedure, we provide a placement constraint for each component of TDCs (i.e. LUTs and CARRY4s) using the .xdc constraint file to specify the location of each sensor. We use Xilinx Vivado 2020 to compile the design into the bitstream file and program the FPGA board. We note that in our experiments, PAD-Lock's AD module is implemented on the PC.

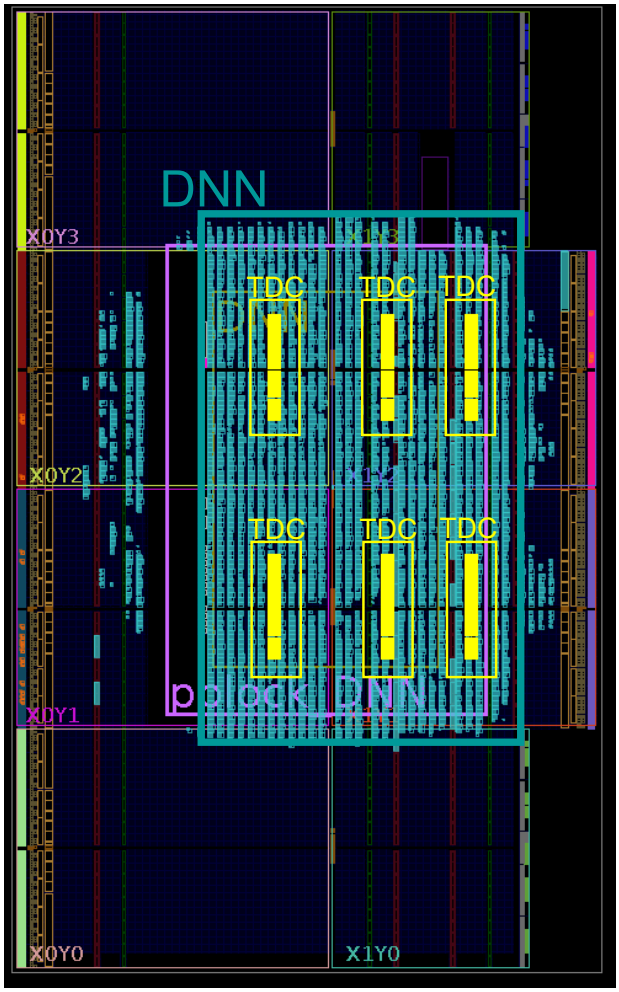
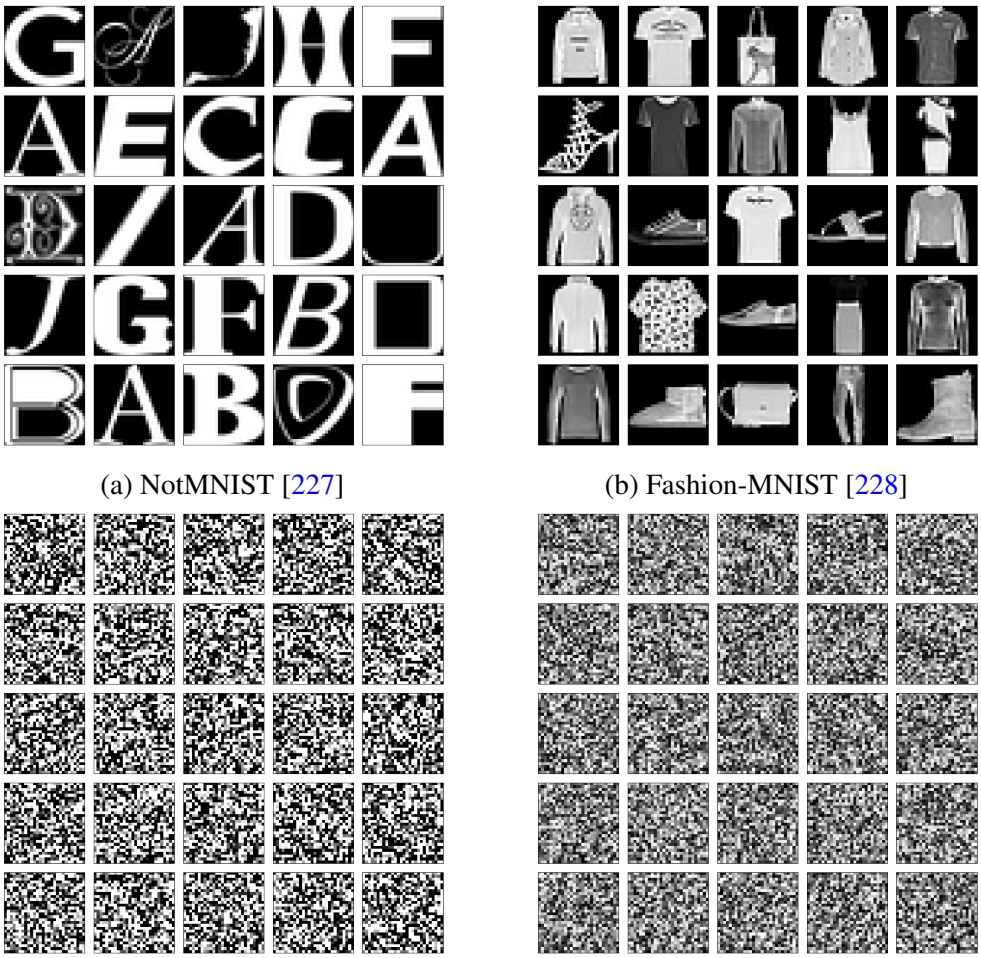


Figure 10.3: The floorplan of FPGA in our experiments. The power consumption of the model during the inference operation is measured by 6 TDC sensors (shown in yellow) scattered across the DNN implementation (indicated in cyan).

However, both the PSC2 and AD modules may be deployed on an FPGA.

For detecting OOD samples, an auto-encoder with three layers, including the latent variable layer was trained on the inference power traces collected for training samples of the MNIST dataset. We used the LeakyReLU [225] function with alpha set to 0.3 for the activation function of the latent variable layer, and the Tanh function for the output layer as the power traces have been scaled and centered around zero. We set the size of the latent variable layer to 50. The mean-squared error (MSE) was adopted as the cost function and the adaptive moment estimation



(a) NotMNIST [227]

(b) Fashion-MNIST [228]

(c) Gaussian Noise

(d) Uniform Noise

Figure 10.4: Some samples from the considered OOD datasets.

(ADAM) [226] technique with the learning rate of 0.001 was used for training the encoder. In our experiments, we measure the reconstruction error between a set of ID validation samples and their reconstructions and set the detection threshold as the 95th percentile of the computed errors.

For detecting adversarial examples, we train a checker model on the inference power traces collected for training samples of the MNIST dataset. The checker model has two hidden layers of 200 and 50 neurons and a softmax output layer of size 10. We used categorical cross-entropy loss and ADAM optimization with a learning rate of 0.001 to train the checker model.

Table 10.1: OOD detection performance using PAD-Lock.

	OOD Dataset			
TNR at 95% TPR ↑	NotMNIST	Fashion-MNIST	Gaussian Noise	Uniform Noise
	100%	96.5%	100%	100%

10.5 Preliminary Results and Future Directions

In this section, we assess the performance of PAD-Lock in detecting OOD and AE data. In our evaluations, the True Positive Rate (TPR) is defined as the ratio of input samples that are from ID and are correctly identified as ID, and the True Negative Rate (TNR) is defined as the ratio of OOD/AE samples that are correctly classified as anomalies by PAD-Lock. We report the performance of our method, using "TNR at 95% TPR" which is commonly used in similar studies [229].

OOD Detection: Our method is evaluated using several OOD datasets namely Fashion-MNIST [228], NotMNIST [227], Gaussian noise, and uniform noise datasets. The Fashion-MNIST dataset contains grayscale images of items from ten types of clothing. NotMNIST is a dataset of grayscale images of letters with various fonts, which is frequently used as an ODD for MNIST classifiers. A Gaussian noise dataset is generated by sampling pixels from a random Gaussian distribution of $\mathcal{N}(0.5, 1.0)$, clipped to [0,1]. Similarly, a uniform noise dataset is generated by sampling each pixel from a uniform distribution between [0,1]. In Figure 10.4, a few examples of considered OOD datasets are presented.

Table 10.1 shows the results for OOD detection using PAD-Lock. To conduct the experiment, we randomly selected 1000 samples from each OOD dataset and measured the TNR of PAD-Lock. As can be seen in Table 10.1, PAD-Lock can offer high precision in detecting OOD

Table 10.2: AE detection performance using PAD-Lock.

	AE Attack	
TNR at 95% TPR ↑	FGSM	PGD-L2
	60.5%	64.51%

samples. The results suggest that the inference power of OOD samples follows a different distribution from the ID samples, thus resulting in high reconstruction errors by the auto-encoder. This enables PAD-Lock to detect OOD samples with high precision.

AE Detection: To detect AEs, the checker model is used to cross-validate the labels predicted by the base classifier, and samples for which the base classifier and checker model predict different labels are considered anomalous. In our experiments, the checker model was trained using the inference power traces collected for MNIST training samples, and it was able to achieve a 95.4% accuracy on the inference power of MNIST testing samples. As a result, AE detection with PAD-Lock achieves a TPR higher than 95%.

In our experiments, we assumed the attacker has full knowledge of the classifier and uses white-box attacks such as (a) Projected Gradient Descent (PGD) with L2 norm [135], and (b) Fast Gradient Sign Method (FGSM) [18] to generate AEs. We generate 1000 AEs using each attack and report the TNR of PAD-Lock in Table 10.2. As can be seen, PAD-Lock can detect more than 60% of all AEs by just analyzing the power consumption of the DNN inference operation.

Future Directions and Discussion: Our preliminary results provide two strong pieces of evidence that DNN's power consumption during inference operation is highly data-dependent, that is, it strongly correlates with the content of input data. First, we observed that an auto-encoder trained on the inference power of ID data performs poorly on OOD samples and causes high reconstruction errors. This indicates that much like OOD and ID data, their inference power

follows different distributions. Second, our checker model could classify the inference power of test samples into their ground truth labels with 95.4% accuracy, demonstrating how sensitive DNN's power consumption is to the input content; even different classes within a dataset have distinct inference power signatures.

Regarding OOD detection with PAD-Lock, while preliminary results are certainly promising, more experiments need to be conducted. We suggest extending experimental evaluations to more advanced DNNs (e.g ResNets, CNNs) and complex datasets. However, that may be challenging as the current setup based on NEXYS 4 DDR cannot run larger DNNs, and thus, simulations must be exported to more sophisticated FPGAs such as Pynq-Z1. Furthermore, we suggest comparing PAD-Lock's performance with contemporary OOD detection methods that operate under similar assumptions to assess its contribution to the state-of-the-art. Additionally, several studies [133] have proposed methodologies to generate OOD samples to fool DNN. The performance of PAD-Lock should be compared against those proposals.

Although preliminary results for AE detection were encouraging, they were not as satisfactory as those for OOD detection. It is important to investigate the reasons behind this phenomenon. In PAD-Lock, we used a checker model to detect AEs as we speculated the inference power of an AE with target label y would be different than that of clean inputs from label y. We suggest conducting more experiments in order to verify this speculation. One way to further examine this assumption could be to use deep generative models such as autoencoders to project the inference power of samples to a low-dimensional feature representation space (where the distance between data points is more meaningful) and then, compare AEs to their neighboring training points in the feature space. This way, if the majority of neighboring points for each AE did not belong to its adversarial label class, it may be concluded that (a) the speculation was correct, that is, the

inference power of adversarial and clean inputs (classified into the same label) differ significantly, and (b) using a checker model to take advantage of this property wasn't the best approach, and alternative techniques need to be investigated.

Moreover, we can think of another direction to improve the performance of PAD-Lock for both OOD and AE detection. TDCs are known to be highly sensitive to environmental noise, such as changes in room temperature. This is to the extent that some studies have proposed placing FPGAs inside refrigerators to minimize the impact of temperature variations on TDC readings [230]. We suspect that environmental noise on the collected TDC measurements may have masked useful patterns that could be used to detect anomalous inputs. We suggest repeating the same experiments using alternative power side-channel collection techniques (e.g. using oscilloscopes) or by using simulated power side channels to investigate this suspicion.

10.6 Summary

In this chapter, we propose PAD-Lock, a novel anomaly detection framework compatible with black-box DNN classifiers. PAD-Lock takes advantage of the power side-channel information of DNNs in order to gain insight into their inner computations and uses this information to detect anomalies. A preliminary analysis of PAD-Lock showed promising results in detecting AEs and especially OODs. We believe PAD-Lock could be an effective and practical anomaly detection framework for commercial DNNs that are often marketed as black-box oracles, but further research is needed to draw a definite conclusion.

Chapter 11: Conclusions and Future Works

11.1 Scope and Contributions of the Thesis

This thesis sets out to address major concerns raised in the semiconductor and AI IP markets, with the goal of encouraging more suppliers and consumers to participate in the market, and consequently, boosting competition, innovation, and growth in both sectors.

IP infringement is widely considered as one of the most serious concerns for semiconductor and AI IP vendors. We propose technical solutions to fortify the protection of silicon and ML IPs against infringements. This is accomplished by presenting two IP watermarking techniques which can assist IP vendors in detecting IP infringements and proving ownership of the pirated content. The first technique was outlined in Chapter 3, where a new dynamic watermarking framework for silicon IPs is discussed. The proposed scheme is based on embedding temperature-sensitive polymorphic gates, evolved with a genetic algorithm, into the gate-level netlist of circuits. This method offers an interesting and unique method for watermark verification; to reveal the ownership information, the circuit must be heated so that polymorphic gates can change logic, allowing the circuit to transition to a special mode and produce the watermark information on the primary outputs. Experimental results showed that the proposed framework can embed large watermark payloads in circuits of various sizes with negligible overheads, making it a practical tool for establishing ownership of silicon IPs. The second watermarking technique

is described in Chapter 4, which introduces GradSigns, a new framework for watermarking DNN classifiers. With GradSigns, the watermark information is embedded by imposing a statistical bias on the gradients of the training cost function with respect to the model's input. Experiments using DNNs trained for various image classification tasks revealed that, unlike contemporary watermarking methods, GradSigns can embed robust multi-bit signatures, which is crucial for a strong proof of ownership. Additionally, experimental results demonstrated that GradSigns is highly loyal to the watermarked model and has little or no impact on its testing accuracy, which is a must-have property for any effective DNN watermarking scheme.

To further improve the protection of IPs against infringements, two IP fingerprinting techniques are presented to help IP vendors track the infringer (dishonest buyer) and hold them responsible. The first technique was discussed in Chapter 6, in which polymorphic gates were used to fingerprint silicon IPs. In the proposed scheme, standard library cells holding SDC conditions are replaced with signal-controlled polymorphic gates, and the value of the fingerprint for each copy is determined by the configuration of the underlying polymorphic gates. Experiments demonstrated that the proposed scheme preserves the functionality of the protected circuit, incurs about half the overhead of similar methods, and can produce unique fingerprints for a large number of copies with negligible embedding costs, which is crucial for an effective fingerprinting framework. Additionally, an analysis was conducted to demonstrate the robustness of the proposed method against various attacks. In Chapter 7, another fingerprinting technique is presented that exploits the testing infrastructures in SoC designs to assign a unique fingerprint to each device by changing the way scan cells are connected in the underlying RSN. To the best of our knowledge, this work is the first to repurpose RSNs for IP protection purposes. The proposed scheme has negligible overheads and requires no extra hardware to embed the fingerprint. Experiments using SIB-based

RSNs implemented for ITC'02 SoC benchmarks demonstrated that this framework can generate unique fingerprints for large numbers of copies - orders of magnitude more than needed for most real-life applications, which is expected from an effective fingerprinting system.

Another equally important problem this thesis addresses is IP buyers' concerns regarding the integration of third-party IPs into their products. We discuss how the black-box commercialization of ML models can complicate efforts to ensure their safety and security from model tampering attacks and anomalous inputs, and why new tamper detection and anomaly detection techniques specifically designed for black-box models are needed. To address this problem, two contributions are made. The first contribution is a new black-box tamper detection technique, called AID, which is outlined in Chapter 9. The proposed method generates a set of test cases that can help verify whether a model has been compromised over API access to the black-box DNN classifier. In AID, the test generation procedure is constructed as an optimization problem in which each test case is optimized to increase the sensitivity of the model's top-1 label to its parameters and maximize the portion of parameters that can be validated by each test case. Evaluations show that the proposed method is highly effective and efficient in detecting a wide selection of model tampering attacks, and offers a major improvement over the state-of-the-art methods. The second contribution is a novel anomaly detection called Pad-Lock, which is discussed in Chapter 10. Pad-lock uses the power side-channel information during DNN inference as a proxy for the model's inner state and discovers patterns that can be used to identify anomalous inputs such as adversarial and out-of-distribution samples. The preliminary analysis of the proposed method shows promise, especially in detecting out-of-distribution samples. Pad-Lock could be an effective and practical anomaly detection framework for commercial DNNs and deserves further investigation.

11.2 Future Work on ML IP Watermarking and Fingerprinting

In this section, we describe four research topics and directions that we recommend for future exploration. We focus on ML IP protection, since it is a relatively new area, whereas semiconductor IP protection is much more mature, and there are recent attempts to integrate existing watermarking and fingerprinting techniques into commercial design tools.

- **Watermarking as a defense against model stealing attacks:** As discussed in Chapter 1, many AI companies market their proprietary ML models as black-box oracles following on-cloud and on-premise business models. In both cases, the model can be accessed through a paid API that allows customers to query the model and retrieve its inference response while keeping the model's internal details (such as its architecture and parameters) hidden. While limiting customers' access to a black-box API may reduce risks of IP infringements, it may not fully eradicate them. In fact, recent studies have shown that ML models behind prediction APIs remain vulnerable to IP infringements through *model stealing attacks* [202]. It has been demonstrated that adversaries can iteratively query a model to extract enough information to train a counterfeit model with comparable performance [202, 231, 232, 233, 234, 235, 236, 237].

  Although the security or robustness against various attacks is one of the most important metrics for any ML IP watermarking technique, it remains unclear whether existing watermarking schemes can survive the vigorous model stealing attacks. It will be interesting to study which contemporary ML IP watermarking techniques can transfer its watermark into counterfeit models and if the transferred watermark will be convincing enough to identify counterfeit

models for legal action against the responsible party.

- **Watermarking machine learning models beyond classification DNNs:** Over the years, several studies have proposed watermarking schemes for DNNs [1, 5, 6, 54, 56, 57, 58, 62, 65, 67, 70, 71]. Nevertheless, it can be observed that the majority of these works were mainly concerned with watermarking classification models, specifically image classification models. Consequently, watermarking techniques for other ML models such as regression models, reinforcement learning models, machine translation models, and so forth have not been sufficiently studied. An interesting future research direction would be to investigate the applicability of GradSigns, our proposed watermarking scheme, to such ML models.

  The GradSigns framework embeds the watermark on the gradient of the cross-entropy cost function (which is the training cost function for classification) with respect to input to the model. While in our experiments we deployed GradSigns for classification tasks, the framework could technically be applied to other learning tasks by replacing the cross-entropy function with a training cost function specific to those tasks. For instance, GradSigns could potentially be used for watermarking regression models by embedding ownership information into the gradient of the mean-squared-error cost function (training cost function for regression tasks) with respect to model inputs. We believe investigating this direction would be a worthwhile endeavor.

- **Fingerprinting machine learning IPs:** Unlike IP watermarking, the concept of fingerprinting has barely been studied for machine learning IPs. To the best of our knowledge, only one fingerprinting method has been proposed for ML models. Chen et al. [238] proposed the DeepMarks fingerprinting framework, which generates a unique binary code vector

for each user as a fingerprint and encodes it into the probability density function of the DNN parameters. However, DeepMarks requires access to model parameters to extract the fingerprint, which makes it inapplicable for commercial models marketed as black-box oracles. Clearly, there is an urgent need for fingerprinting techniques suitable for commercial ML models, and more effort should be put into this area. It would be worthwhile to investigate whether the information hiding techniques originally proposed for watermarking DNNs can also be used for fingerprinting them as well.

We believe any existing watermarking scheme that meets the following criteria can also be used to fingerprint ML IPs: (a) ultra-low embedding costs: techniques that embed ownership information during model training are not practical for embedding fingerprints as it is unreasonable to expect model vendors to repeat the entire training process every time they want to create a fingerprinted copy. In this sense, only methods that embed information after model training can potentially be used for fingerprinting purposes; (b) robustness to collusion attack: any method considered for fingerprinting should be robust against collusion attacks. As discussed in Chapter 5, collusion attacks are malicious attempts in which customers who have access to copies with different fingerprints collude to construct a model with no (or invalid) fingerprints; (c) scalability: a potential candidate for fingerprinting should be capable of realizing unique fingerprints for a large body of customers.

- **Side-channel based watermarking of DNNs:** Future research should examine the possibility of using power side channel analysis for copyright protection of DNNs. The main idea is to implement a communication channel within the power side-channel of DNN implementations to transmit the owner's watermark. In such schemes, detecting a watermark requires

measuring the power consumption of the device that supposedly hosts the stolen DNN and then performing a power analysis to correlate the value of the watermark with the captured power readings. If the watermarked DNN is deployed on the device, the analysis will reveal the presence of the inserted watermark.

One might wonder how the proposed approach improves upon state-of-the-art? It can be explained as follows: existing watermarking techniques assume that the model vendor can send arbitrary inputs to the model under investigation and observe its output to extract and verify the watermark information. However, this assumption is not valid for models that are integrated into complex systems. With side-channel based watermarking techniques, vendors can detect and verify embedded watermarks without controlling or observing the model's inputs and outputs, simply by acquiring power measurements of the system. This property makes side-channel based watermarking a superior approach to existing techniques, thus making it a promising area for further investigation.

Appendix A: List of Publications

1. **Aramoon, Omid**, Gang Qu, and Aijiao Cui. ”Building Hardware Security Primitives Using Scan-based Design-for-Testability.” 2022 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2022.

2. **Aramoon, Omid**, Pin-Yu Chen, Gang Qu, and Yuan Tian. ”Meta Federated Learning.” arXiv preprint arXiv:2102.05561 (2021).

3. **Aramoon, Omid**, and Gang Qu. ”Provably Accurate Memory Fault Detection Method for Deep Neural Networks.” In Proceedings of the 2021 on Great Lakes Symposium on VLSI, pp. 443-448. 2021.

4. **Aramoon, Omid**, Pin-Yu Chen, and Gang Qu. ”Don’t Forget to Sign the Gradients!.” Proceedings of Machine Learning and Systems 3 (2021): 194-207.

5. Dunlap, Timothy, **Omid Aramoon**, Gang Qu, Tian Wang, Xiaoxin Cui, and Dunshan Yu. ”A Novel Circuit Authentication Scheme Based on Partial Polymorphic Gates.” In 2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 1-4. IEEE, 2021.

6. Tan, Benjamin, Siddharth Garg, Ramesh Karri, Yuntao Liu, Michael Zuzak, Abhisek Chakraborty, Ankur Srivastava, **Omid Aramoon**, Qian Xu, Gang Qu, Adam Porter, Jeno Szep, and

Warren Savage. 2021. ”Invited: Independent Verification and Validation of Security-Aware EDA Tools and IP.” In 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 1299-1302. IEEE, 2021.

7. **Aramoon, Omid**. “Do You Sign Your Model?.” International Conference on Machine Learning 2020 Workshop on ”Challenges in Deploying and monitoring Machine Learning Systems” (DMMLSys-20).

8. **Aramoon, Omid**, and Gang Qu. ”Impacts of machine learning on counterfeit IC detection and avoidance techniques.” In 2020 21st International Symposium on Quality Electronic Design (ISQED), pp. 352-357. IEEE, 2020.

9. Wang, Qian, **Omid Aramoon**, Pengfei Qiu, and Gang Qu. ”Efficient transfer learning on modeling physical unclonable functions.” In 2020 21st International Symposium on Quality Electronic Design (ISQED), pp. 1-6. IEEE, 2020.

10. Chen, Xi, **Omid Aramoon**, Gang Qu, and Aijiao Cui. ”Balancing testability and security by configurable partial scan design.” In 2018 IEEE International Test Conference in Asia (ITC-Asia), pp. 145-150. IEEE, 2018.

11. **Aramoon, Omid**, Xi Chen, and Gang Qu. ”A reconfigurable scan network based IC identification for embedded devices.” In 2018 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 479-484. IEEE, 2018.

12. Wang, Tian, Xiaoxin Cui, Dunshan Yu, **Omid Aramoon**, Timothy Dunlap, Gang Qu, and Xiaole Cui. ”A novel polymorphic gate based circuit fingerprinting technique.” In Proceedings of the 2018 on Great Lakes Symposium on VLSI, pp. 141-146. 2018.

13. Wang, Tian, Xiaoxin Cui, Dunshan Yu, **Omid Aramoon**, Timothy Dunlap, Gang Qu, and Xiaole Cui. ”Polymorphic gate based IC watermarking techniques.” In 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 90-96. IEEE, 2018.

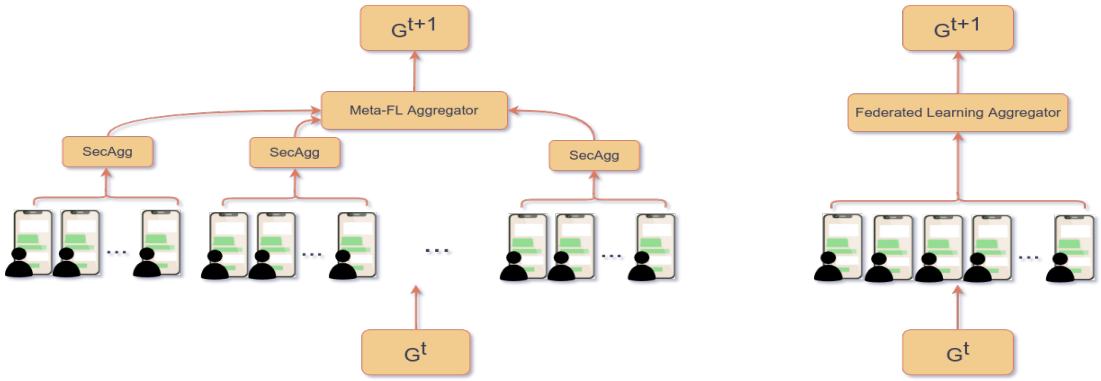
Appendix B: Meta Federated Learning

In this appendix, we include one of our publications [239] that did not fall within the scope of the dissertation.

B.1 Introduction

Federated Learning (FL) is a distributed learning framework that enables millions of clients (e.g., mobile and edge devices) jointly train a deep learning model under the supervision of an orchestration server [240, 241, 242]. Taking advantage of training data distributed among the crowd of clients enables federated learning to train a highly accurate shared global model. Federated learning has gained significant interest from the industry with many tech companies, including Google and Apple, deploying this framework to improve their services, such as next word prediction for messaging on mobile devices and voice recognition for digital assistants [243].

In every round of federated learning, the central server randomly selects a cohort of participants to locally train the joint global model on their private data and submit an update to the server, which would be aggregated into the new global model. Federated learning decouples model training from the need to access participants' training data by collecting focused model updates that contain enough information for the server to improve the global model without revealing too



(a) Meta federated learning (b) Federated learning

Figure B.1: Overview of model training in baseline and meta federated learning.

much about the client's private data [243].

While collecting model updates, instead of centralizing raw training data, significantly reduces privacy concerns for participating clients, it does not offer any formal privacy guarantees. Recent studies have shown that model updates can still leak sensitive information about the client's data [244, 245], which proves that preserving the privacy of clients is only a promise, and certainly not the reality of federated learning.

To systematically address such privacy concerns, recent FL settings deploy Secure Aggregation (SecAgg) [246], a cryptographic protocol that enables the server to compute the aggregate of updates and train the global model while keeping each individual update uninspectable at all time. Looking from the server's point of view, secure aggregation can be a "double-edged sword." On the one hand, it can systematically mitigate privacy risks for participants, which would make federated learning more appealing to clients and eventually result in higher client turnout. On the other hand, it would facilitate training time adversarial attacks by masking participants' contributions.

Training time adversarial attacks may have targeted [247, 248, 249] or untargeted adversarial

objectives [250, 251]. In untargeted attacks, the adversary aims to corrupt the learned model so that it would perform poorly on the learning task at hand. However, in targeted attacks, the adversary's goal is to force the model to learn certain adversarial sub-task in addition to the primary learning task. Targeted attacks are harder to detect compared to untargeted attacks as the adversary's objective is unknown. Perhaps the most prevalent example of targeted attacks is backdoor attacks, which have been extensively explored for the centralized learning settings [249, 252, 253]. In backdoor attacks, the adversary's goal for the learned model is to misclassify inputs containing certain triggers while classifying inputs without the trigger correctly.

Known techniques in mitigating backdoor attacks in the centralized setting are not applicable to federated learning. Successful defenses such as data sanitization [254] and network pruning [255] require careful examination of clients' training data or access to a proxy dataset with similar distribution as the global dataset. None of these requirements hold in federated learning.

Moreover, contemporary defenses against backdoor attacks in FL require examination of participant's model updates, which is not compatible with secure aggregation. Even in the absence of secure aggregation, inspecting the clients' updates is not acceptable due to privacy concerns and regulations.

This study seeks to answer the following question, *" Is it possible to defend against backdoor attacks when secure aggregation is in place?"*, a question that has not been investigated by prior studies. To this end, we propose Meta Federated Learning (Meta-FL), a novel federated learning framework that not only preserves the privacy of participants but also facilitates defense against backdoor attacks.

In our framework, we take full advantage of the abundance of participants by engaging more than one training cohorts at each round to participate in model training. To preserve the

privacy of participants, Meta-FL bootstraps the SecAgg protocol to aggregate updates from each training cohort. In Meta-FL, the server is provided with a set of cohort aggregates, instead of individual model updates, which are further aggregated to generate the new global model. Figure B.1a illustrates the overview of model training in Meta-FL.

Meta-FL moves defense execution point from update level to aggregate level which facilitates mitigating backdoor attacks by offering the following advantages: (i) server can monitor cohort aggregates without violating the privacy of participants. Therefore, the adversary is forced to be mindful of their submissions and maintain stealth on the aggregate level as aggregates that are statistically different from others are likely to get flagged and discarded; (ii) cohort aggregates exhibit less variation compared to individual client updates, which makes it easier for the server to detect anomalies, and (iii) adversary faces competition from benign clients to hold control of the value of cohort aggregates which hinders them from executing intricate defense evasion techniques.

Our key contributions can be summarized as follows:

- We propose Meta federated learning, a novel federated learning framework that facilitates defense against backdoor attacks while protecting the privacy of participants.
- We show that moving the defense execution point from individual update level to aggregate level is effective in mitigating backdoor attacks without compromising privacy.
- We perform a systematic evaluation of contemporary defenses against backdoor attacks in both standard federated learning and Meta-FL. Results on two classification datasets: SVHN [102] and GTSRB [207], show that Meta-FL enhances contemporary defense performance in terms of robustness to adversarial attacks and utility.

B.2 Background

B.2.1 Federated Learning

Federated learning is a machine learning setting that enables millions of clients (mobile or edge devices) to jointly train a deep learning model using their private data without compromising their privacy. The training procedure in federated learning is orchestrated by a central server responsible for providing the shared global model to participants and aggregating their submitted model updates to generate the new global model. The key appeal of federated learning is that it does not require centralizing participating users' training data, which makes it ideal for privacy-sensitive tasks.

A standard FL setting consists of P participating clients. Each client i holds a shard of training data D_i which is private to the client and is never shared with the orchestration server. In each round t of federated learning, the central server randomly selects a set ζ^t of c clients, and broadcasts the current global model G^t to them. Selected set of clients ζ_t is referred to as *training cohort* of round t. Each client i in the training cohort locally and independently trains the joint model G^t using Stochastic Gradient Descent (SGD) optimization algorithm for E epochs on its local training data D_i to obtain a new local model L_i^{t+1}, and submits the difference $L_i^{t+1} - G^t$ as its model update to the central server. Next, the central server averages model updates submitted by clients in the training cohort and updates the shared global model using its learning rate η to obtain the new global model G^{t+1}, as shown in Equation B.1. Model training resumes until the global model converges to an acceptable performance, or certain training rounds are completed.

$$G^{t+1} = G^t + \frac{\eta}{n} \sum_{i=i}^{n} (L_i^{t+1} - G^t) \tag{B.1}$$

B.2.2 Secure Aggregation

Secure Aggregation (SecAgg) [246] is a secure multi-party computation protocol that can reveal the sum of submitted model updates to the server (or aggregator) while keeping each individual update uninspectable at all times. Secure Aggregation consists of three phases, *preparation*, *commitment* and *finalization* [256]. In the preparation phase, shared secrets are established between the central server and participating clients. Model updates from clients who drop out during the preparation phase will not be included in the aggregate. Next, in the commitment phase, each device uploads a cryptographically masked model update to the server, and the server computes the sum of the submitted mask updates. Only clients that successfully commit their masked model updates will contribute to the final aggregate. Lastly, in the finalization phase, committed clients reveal sufficient cryptographic secrets to allow the server to unmask the aggregated model update.

B.2.3 Robust Aggregation Rules and Defenses

Numerous studies have proposed robust aggregation rules [250, 257, 258] to ensure convergence of distributed learning algorithms in the presence of adversarial actors. The majority of studies in this line of work assume a byzantine threat model in which the adversary can cause local learning procedures to submit any arbitrary update to ensure convergence of learning algorithms to an ineffective model. In addition to robust aggregation rules, several works have proposed

novel defenses [259, 260] against backdoor and poisoning attacks in federated learning. In what follows, we review several of the techniques which we experiment with in Section B.5.

Krum. The Krum algorithm, proposed by [250], is a robust aggregation rule which can tolerate f byzantine attackers out of n participants selected at any training round. Krum has theoretical guarantees for the convergence should the condition $n \geq 2f + 3$ hold true. At any training round, for each model update δ_i, Krum takes the following steps: (a) computes the pairwise euclidean distance of $n - f - 2$ updates that are closest to δ_i, (b) computes the sum of squared distances between update δ_i and its closest $n - f - 2$ updates. Then, Krum chooses the model update with the lowest sum to update the parameters of the joint global model.

Coordinate-Wise Median. In Coordinate-Wise Median (CWM) aggregation rule [257], for each j_{th} model parameter, the j_{th} coordinate of received model updates are sorted, and their median is used to update the corresponding parameter of the global model.

Trimmed Mean. Trimmed Mean (TM) is a coordinate wise aggregation rule [257]. for $\beta \in [0, \frac{1}{2})$, trimmed mean computes the j_{th} coordinate of aggregate of n model updates as follows: (a) it sorts the j_{th} coordinate of the n updates, (b) discards the largest and smallest β fraction of the sorted updates, and (c) takes the average of remaining $n(1 - 2\beta)$ updates as the aggregate for the j_{th} coordinate.

RFA. RFA [258] is a robust privacy-preserving aggregator which requires a secure averaging oracle. RFA aggregates local models by computing an approximate of the geometric median of their parameters using a variant of the smoothed Weiszfeld's algorithm [261]. RFA appears to be tolerant to data poisoning attacks but can not offer byzantine tolerance as it still requires clients to compute aggregation weights according to the protocol. Relying on clients to follow a defensive protocol without a proper means to attest to the correctness of computations on the client-side

cast doubts on the practicality of RFA. To the best of our knowledge, RFA is the only existing defense that is compatible with secure aggregation.

Norm Bounding. Norm Bounding (NB) is an aggregation rule proposed by [260], which appears to be robust against false-label backdoor attacks. In this aggregation rule, a norm constraint M is set for model updates submitted by clients to normalize the contribution of any individual participants. Norm bounding aggregates model updates as follows: (a) model updates with norms larger than the set threshold M are projected to the $l2$ ball of size M and then (b) all model updates are averaged to update the joint global model.

Differential Privacy. Differential Privacy (DP) originally was designed to establish a strong privacy guarantee for algorithms on aggregate databases, but it can also provide a defense against poisoning attacks [262, 263]. Extending DP to federated learning ensures that any participant's contribution is bounded and therefore, the joint global model does not over-fit to any individual update. DP is applied in FL as follows [243]: (a) server clips clients' model update by a norm M, (b) clipped updates are aggregated, then (c) a Gaussian noise is added to the resulted aggregate. DP has recently been explored and shown to be successful against false-label backdoor attacks in a study by [260].

B.3 Problem Definition and Threat Model

In this section, we present the objectives, capabilities, and schemes of backdoor attackers that are commonly used in prior studies. In other words, our proposed Meta-FL framework does not make any additional assumptions.

Attacker's Objective. Similar to prior arts such as [264, 265], we consider an adversary

whose goal is to cause misclassifications to a targeted label T for inputs embedded with an attacker-chosen trigger. As opposed to Byzantine attacks [250], whose purpose is to make the learning algorithm converge to a sub-optimal or utterly ineffective model, the adversary's goal in backdoor attacks is to ensure that the joint global model achieves high accuracy on both the backdoor sub-task and the primary learning task at hand.

Attacker's Capability. We make the following assumptions about the attacker's capabilities: (a) we assume the attacker controls a number of participants, which are referred to as *sybils* in the literature of distributed learning. Sybils are either malicious clients which are injected into a federated learning system or benign clients whose FL training software has been compromised by the adversary; (b) following Kerckhoffs's theory [266], we assume a strong attacker who has complete control over local data and training procedure of all its sybils. The attacker can modify the training procedure's hyperparameters and is capable of modifying model updates before submitting them to the central server; (c) the adversary is not capable of compromising the central server or influencing other benign clients, and more importantly, does not have access to benign clients' local model, training data and submitted updates.

Attack scheme. In our evaluations, we consider two backdoor attack schemes which are referred to as "Naive" and "Model Replacement" in literature [265]. In both schemes, adversaries train their local model with a mixture of clean and backdoored data, and model updates are computed as the difference in the parameters of the backdoored local model and the shared global model. In the naive approach, the adversary submits the computed model update as is, while in the model replacement attack, the model update is scaled using a scaling factor to cancel the contribution of other benign clients and increase the impact of the adversarial update on the joint global model. A carefully chosen scaling factor for adversarial updates can guarantee the

replacement of the joint global model with the adversary's backdoored local model.

B.4 Meta Federated Learning

In this section, we first discuss the challenges in mitigating backdoor attacks in federated learning. Then, we propose Meta Federated Learning (Meta-FL), and explain how it improves robustness to backdoor attacks while preserving the privacy of participating clients.

Challenges in defending against backdoor attacks in federated learning are two-fold:

Challenge 1. Inspecting model updates is off limits with or without secure aggregation. Recent studies have demonstrated that model updates can be used to partially reconstruct clients' training data [249, 252, 253]; therefore, any defensive approach which requires examination of submitted updates is a threat to the privacy of participants, and is against privacy promises of federated learning. Moreover, inspecting model updates simply is not feasible in systems augmented with SecAgg. Privacy promises of federated learning prohibit the server from auditing clients' submissions which gives the adversary the privilege to submit any arbitrary value without getting flagged as anomalous. We refer to this privilege as ***submission with no consequences***.

Challenge 2. Even without the restrictions mentioned above, defending against backdoor attacks would not be a trivial task. Model updates submitted by clients show high variations which makes it extremely difficult for the central server to identify whether an update works toward an adversarial goal. The sporadicity observed from model updates originates from the non-i.i.d distribution of the original dataset among participants, and the fact that each update is the product of stochastic gradient descent, a non-deterministic algorithm whose output is not merely a function of its input data.

Motivated to address the challenges above, we propose, Meta-FL, a novel federated setting that not only protects the privacy of participants but also aids the server in defending against backdoor attacks. Algorithm 6 summarizes different steps of model training in our framework, which we will cover in detail here.

In each round t of training in Meta-FL, central server randomly selects π cohorts $\{\zeta_1^t, \zeta_2^t, ..\zeta_\pi^t\}$, each containing c unique clients (Line 3). Training cohorts can be sampled in-order or independently. In the former case, each cohort is sampled after another, and thus, no client will be a member of more than one cohort ($\zeta_i^t \cap \zeta_j^t = \emptyset$). In the recent case, there is no inter-dependency among cohort selection, and therefore, cohorts can have clients in common; this scenario is more suitable for cases where ($P \leq \pi c$). Next, server broadcasts global model G^t to clients in each cohort (Line 5), each client i locally and independently trains the model G^t on their local training data to obtain a new local model L_i^{t+1} , and compute their model update δ_i as $L_i^{t+1} - G^t$ (Line 7). Then, the server establishes π separate instances of SecAgg protocol to concurrently compute the aggregate of updates submitted from clients of each cohort (Line 9). Finally, in the last stage of training in Meta-FL, the central server aggregates the "cohort updates" using aggregation rule Γ, and updates the joint model with its learning rate η to obtain the next shared global model G^{t+1}, as shown in Line 11 of Algorithm 6.

In our framework, plain model updates never leave the client's side. All participants are required to follow the SecAgg protocol and submit cryptography masked updates. SecAgg guarantees that the server is able to aggregate the masked submissions to update the global model but can not obtain the value of each individual update. While each cohort aggregate may still leak information about collective training data of cohort members, the inferred information can not be associated with any individual client; therefore, the privacy of participants is preserved in

Algorithm 6 Meta-FL framework

Initialize shared global model
for each round t in 1,2,3.. **do**
  Select π training cohorts $\{\zeta_1^t, \zeta_2^t, ..\zeta_\pi^t\}$ with $|\zeta_1^i| = c \; \forall i \in \{1, 2, .., \pi\}$.
  for cohort ζ_j^t in $\{\zeta_1^t, \zeta_2^t, ..\zeta_\pi^t\}$ **in parallel do**
    Broadcast global model G^t to cohort members.
    for client i in cohort ζ_j^t **in parallel do**
      $\delta_i^t \leftarrow ClientUpdate(i, G^t)$
    end for
    $\Delta_j^t \leftarrow SecAgg(\delta_1^t, \delta_2^t, .., \delta_c^t)$
  end for
  $G^{t+1} = G^t + \eta \, \Gamma(\Delta_1^t, \Delta_2^t, .., \Delta_\pi^t)$
end for

Meta-FL.

In Meta-FL, as the central server can only see the aggregate of training cohorts, defense mechanisms are obliged to carry out on the aggregate level rather than update level. This property offers the server several advantages in mitigating backdoor attacks, which we will cover in the rest of this section. However, before we can proceed, we need to define several concepts that are essential for understanding what follows.

In the rest of this chapter, we refer to a training cohort as adversarial if and only if there exists at least one malicious client among its members. Naturally, a cohort is referred to as benign if none of its members are malicious. Moreover, we refer to the aggregate of updates from a benign and an adversarial cohort as a benign and adversarial aggregate, respectively.

Moving defense execution point from update level to aggregate level facilitates mitigating backdoor attacks as it offers the following advantages:

Advantage 1. Server is allowed to inspect and monitor cohort aggregates. This property forces the adversary to maintain stealth on the aggregate level as adversarial aggregates which are statistically different from other benign aggregates are likely to get detected and discarded by

the server. Therefore, Meta-FL, revokes the privilege of submission with no consequences for the adversary.

Advantage 2. Cohort aggregates are less sporadic compared to individual client updates which aids the server in detecting anomalies. This advantage takes on the *challenge 2* discussed above. By drawing an analogy to *simple random sampling* in statistics [267], we demonstrate that cohort aggregates show less variation across each coordinate compared to individual updates.

For ease of analysis, we assume that training cohorts are sampled independently meaning that there is no inter-dependency among client selection in each cohort. In this case, at any round t, updates submitted by any cohort of c clients is essentially a random sample of size c collected without replacement from the population of model updates. Assuming that updates are averaged as in Equation B.1, cohort aggregates are in fact *sample means* of the model update population. As the composition of cohorts is a random process, cohort aggregates are thus random variables whose distribution is determined by that of model updates as shown below (for proof refer to [267]):

$$Var(\Delta_j) = \frac{\sigma_j^2}{c}\left(\frac{P-c}{P-1}\right), \quad \mathbb{E}[\Delta_j] = \mathbb{E}[\mu_j] \tag{B.2}$$

Here, σ_j^2 and μ_j denote variance and mean of population of model updates across the j_{th} coordinate, respectively, and Δ_j indicates the j_{th} coordinate of a cohort aggregate Δ. Assuming that each cohort contains more than one client ($1 < c$), it would be trivial to show that $\frac{P-c}{c(P-1)} < 1$. Therefore, we can prove that the variance of cohort aggregates across any coordinate j is upper bounded by the variance of the population of model updates across that coordinate as shown below:

$$Var(\Delta_j) = \sigma_j^2 \left(\frac{P-c}{c(P-1)} \right) < \sigma_j^2 \tag{B.3}$$

A closer look at Equation B.3 reveals that the server can further reduce the variance of cohort aggregates by increasing the size of training cohorts which would cause $\frac{P-c}{c(P-1)}$ to become smaller and closer to zero. Lower variation from cohort aggregates makes it easier for outlier detection-based defenses to infer patterns of the benign observations, and effectively detect malicious instances.

Advantage 3. As adversarial updates are aggregated with other updates, sybils face competition from benign clients to control the value of cohort aggregate. This property makes it harder for the adversary to meticulously arrange values of adversarial aggregates to evade deployed defenses.

Our empirical evaluations in Section B.5 will demonstrate that the advantages mentioned above in fact aid contemporary defense to perform better against backdoor attacks in Meta-FL compared to the baseline federated learning.

B.5 Experimental Evaluation and Discussion

B.5.1 Datasets and Experiment Setup

We study Meta-FL on two classification datasets namely SVHN [102] and GTSRB [207] with non-i.i.d. data distributions. **GTSRB** is s traffic sign dataset with 39,209 training and 12,630 test samples, where each sample is labeled with one of the possible 43 classes, and **SVHN** is a dataset of more than 100k images of digits cropped out of images of house and street numbers. Table B.1 reports the architecture and hyperparameters of benchmark models used for the GTSRB

and SVHN datasets.

We use a Dirichlet distribution with parameter $\alpha = 0.9$ to partition GTSRB and SVHN datasets into disjoint non-i.i.d shards and then distribute them among 150 and 300 clients, respectively. Following a similar setup to prior arts, each participating client trains their local model using SGD for 5 epochs with a batch size of 64 and a learning rate of 0.1. Both Meta-FL and baseline FL resume the training process until a certain number of training rounds are completed. Throughout our experiments, GTSRB and SVHN models are trained for 75 and 50 rounds, respectively.

For all experiments, pixel pattern backdoor attacks are performed in which the adversary aims to influence the model to misclassify inputs from a base label as a target label upon the presence of an attacker's chosen pattern (trigger). We set the adversarial trigger as a white square located at the top left corner of the image which roughly covers 9% of the entire image. The objective of backdoor attacks in GTSRB and SVHN datasets is to mispredict images of "Speed limit 80 miles per hour" as "Speed limit 50 miles per hour" and images of "digit 6" as "digit 1", upon the presence of the white box trigger.

Table B.1: Model architecture for SVHN and GTSRB datasets.

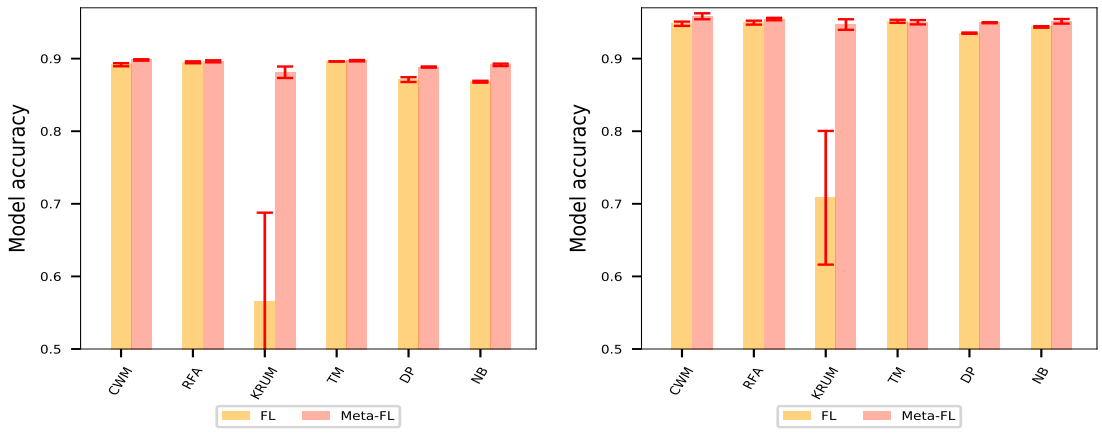
GTSRB		SVHN	
Layer Type	Filter/Unit	Layer Type	Filter/Unit
Conv + ReLU	$3 \times 3 \times 32$	Conv + ReLU	$3 \times 3 \times 32$
Conv + ReLU	$3 \times 3 \times 32$	Conv + ReLU	$3 \times 3 \times 32$
Conv + ReLU	$3 \times 3 \times 64$	Conv + ReLU	$3 \times 3 \times 64$
Conv + ReLU	$3 \times 3 \times 64$	Conv + ReLU	$3 \times 3 \times 64$
Conv + ReLU	$3 \times 3 \times 128$	Conv + ReLU	$3 \times 3 \times 128$
Conv + ReLU	$3 \times 3 \times 128$	Conv + ReLU	$3 \times 3 \times 128$
FC + ReLU	43	FC + ReLU	512
Softmax	43	FC + ReLU	10
		Softmax	10

In the rest of the chapter, we denote each Meta-FL framework by two parameters as **MFL-i-j**, where i and j indicate the number and size of training cohorts, respectively. We also use a similar notation **FL-k** to describe baseline FL systems, where k indicates the size of the training cohort.

Similar to the analysis in [260], we consider fixed frequency attack models to explore a wide range of attack scenarios. In the baseline FL, **Attack-f-k** describes a scenario where k sybils appear at every f rounds of training to mount their attack. For the case of Meta-FL setting, **Attack-f-k** describes the case where at every f round of training, k training cohorts contain an adversarial client.

B.5.2 Utility of Meta-FL

In this section, we compare the utility of Meta-FL against the baseline setting in terms of model accuracy. In this experiment, we evaluate the utility of baseline and Meta-FL frameworks across various FL configurations and aggregation rules. Note that for a fair comparison, we make sure the number of clients participating in each round of model training is equal across both frameworks. Figure B.2 reports the test accuracy of models trained in Meta-FL and baseline settings deploying different defenses and aggregation rules. As reflected, federated training with Meta-FL results in more accurate models compared to the baseline setting. All defenses and aggregation rules offer better utility in our framework. Even the Krum aggregation rule which has been known to cause a large drop in performance of the learned model in baseline FL [265, 268] can train models with comparable performances in Meta-FL.



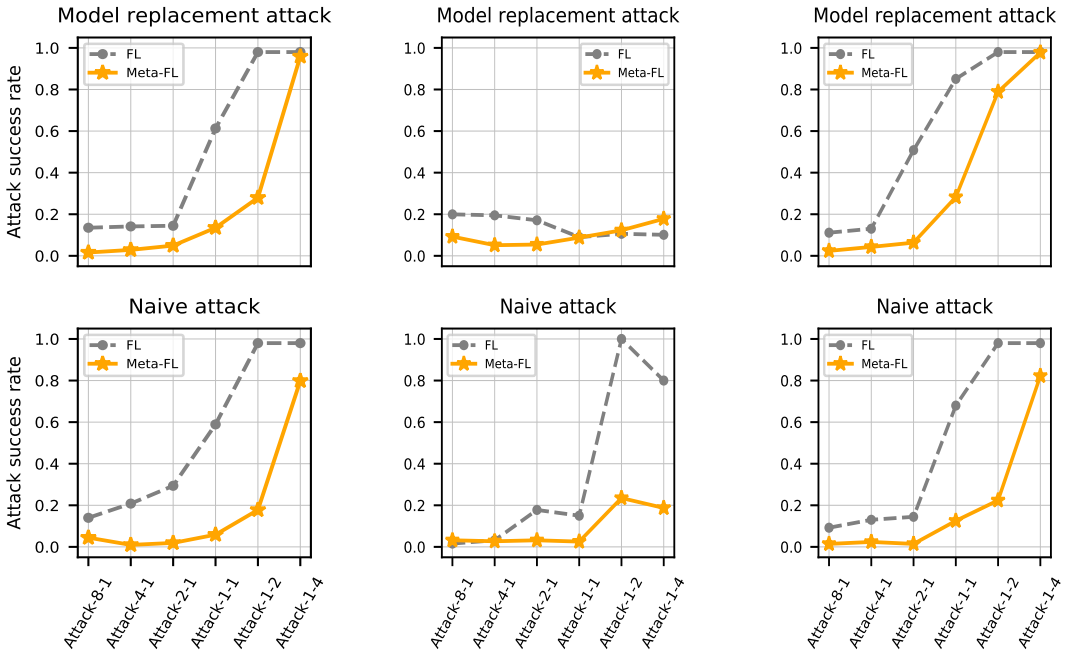
(a) SVHN (b) GTSRB

Figure B.2: Comparing utility of Meta-FL against baseline FL in terms of model accuracy.

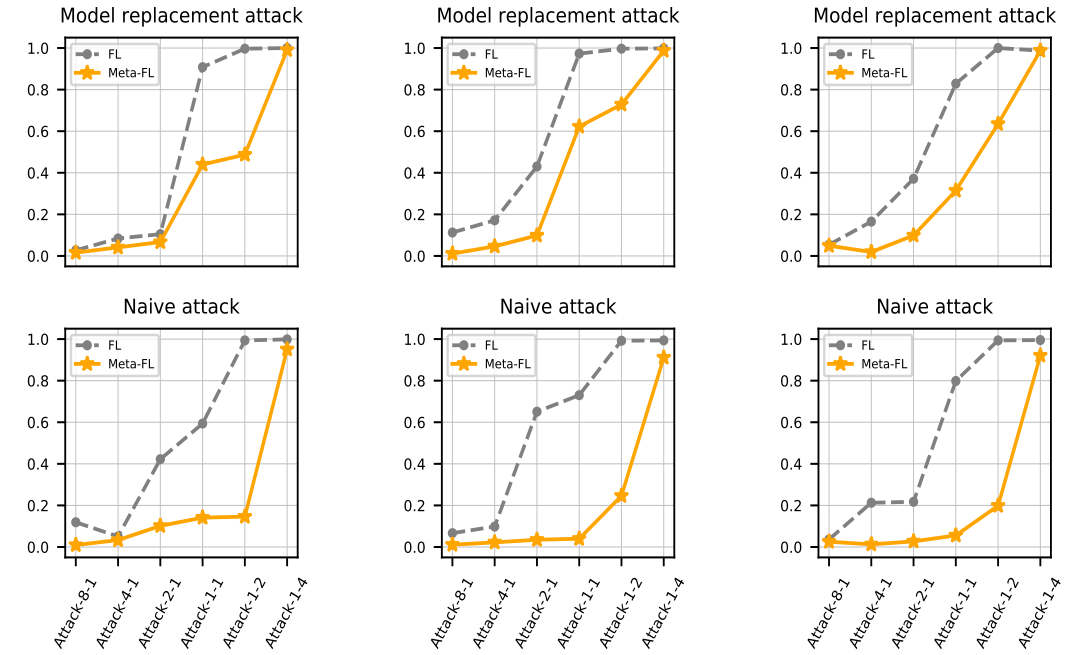
B.5.3 Robustness of Meta-FL

In this section, we systematically compare the capabilities of contemporary defenses against backdoor attacks in both baseline and meta federated learning. Our empirical evaluation in this section shows that all defenses benefit from the advantages discussed in Section B.4 and offer better robustness in Meta-FL.

Figures B.3 and B.4 report performance of contemporary defenses against backdoor attacks on GTSRB and SVHN benchmarks, respectively. We extend our experiments to both Meta-FL (MFL-15-5) and baseline FL (FL-5) frameworks for each dataset. In our evaluations, we consider defenses such as Krum, Coordinate-Wise Median (CWM), trimmed mean (TM), norm bounding (NB), differential privacy (DP) and RFA. Hyperparameter and implementation details of these techniques are as follows: **(1)** For Krum, to meet the convergence condition $n \geq 2f + 3$, we set $f = 6$. **(2)** In Trimmed mean, the parameter β is set to 0.20. **(3)** For RFA, the maximum iteration of the Weiszfeld algorithm and the smoothing factor is set to 10 and 10^{-6}, respectively. **(4)** In norm bounding defense, as the original work [260] did not provide a recipe to decide the norm

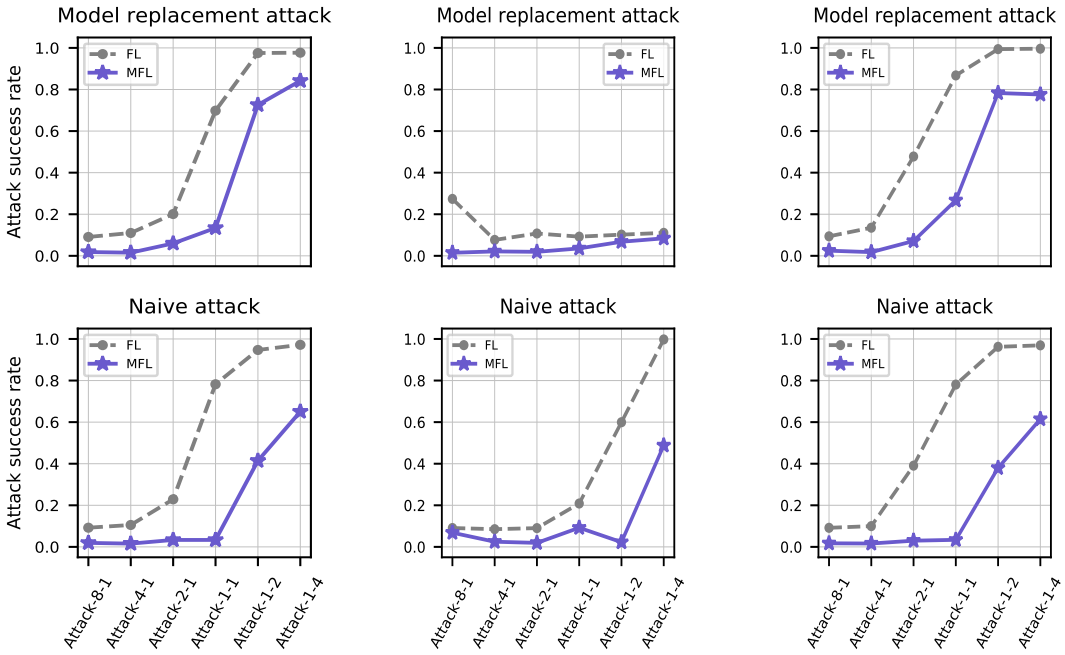


(a) CWM (b) Krum (c) Trimmed Mean

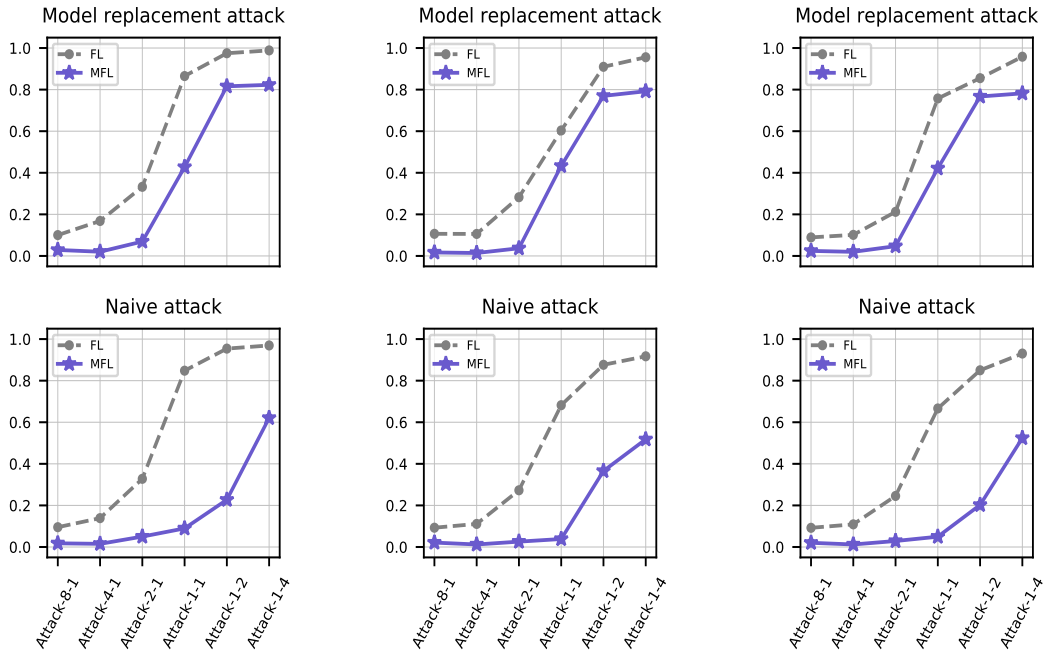


(d) RFA (e) Norm Bounding (f) Diff. Privacy

Figure B.3: Evaluating performance of contemporary defenses against naive and model replacement backdoor attacks on GTSRB model.



(a) CWM (b) Krum (c) Trimmed Mean



(d) RFA (e) Norm Bounding (f) Diff. Privacy

Figure B.4: Evaluating performance of contemporary defenses against naive and model replacement backdoor attacks on SVHN model.

threshold M, we developed our own approach to determine M. In our experiments, at each round, we set the norm threshold M to the norm of the smallest aggregand to ensure all aggregands will have an equal $l2$ norm before aggregation. In federated learning, as the global model converges, model updates (and therefore cohort aggregates) start to fade out and have smaller norms, and thus, setting a constant norm threshold for all training rounds would not be effective, which is why we took a dynamic approach to decide M. **(5)** For differential privacy the hyper-parameter M is set similar to norm bounding and then a Gaussian noise $\mathcal{N}(0.0,\, 0.001^2)$ is added to the aggregate of updates (or cohort aggregates) before updating the global model.

We experiment with several attack scenarios to systematically evaluate the performance of each defense against adversaries with a wide range of resources at hand. As we move along the attack scenarios denoted on the horizontal axis of diagrams in Figures B.3 and B.4, the adversary becomes more and more powerful and appears more frequently with more sybils at each round.

For a fair evaluation of contemporary defense across Meta-FL and baseline FL, we make sure the defender faces similar challenges in both frameworks. Throughout our experiments in this section, we set the number of training cohorts in Meta-FL equal to the number of selected clients in baseline FL to ensure that server sees the same number of aggregands (client updates in baseline FL and cohort aggregates in Meta-FL) across both cases. Moreover, the way our attack scenarios are defined ensures that the same number of aggregands are adversarial across both frameworks.

Across both Meta-FL and baseline FL frameworks, the scaling factor for model replacement attack is set equal to the size of training cohorts to ensure that submissions from adversarial clients survive the averaging procedure and overpower the aggregate of their corresponding cohort. For attack scenarios in which multiple sybils appear in the same round, we assume they coordinate

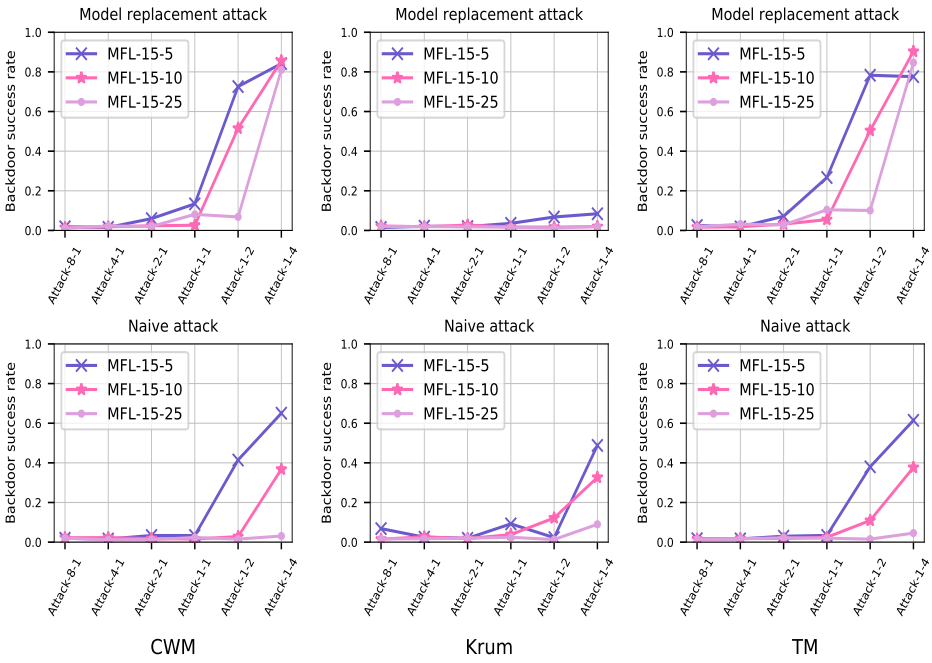
and divide the scaling factor among themselves evenly.

Figures B.3 and B.4 shows that *Meta-FL puts all defense at an advantage in mitigating against backdoor attacks.* Attack success rate of both the naive and model replacement approach in Meta-FL (solid lines) is lower than in baseline FL (dashed lines) when the same defense is in place across both frameworks. Therefore, our empirical evaluations show that existing defenses are more robust to backdoor attacks in Meta-FL compared to baseline FL across.

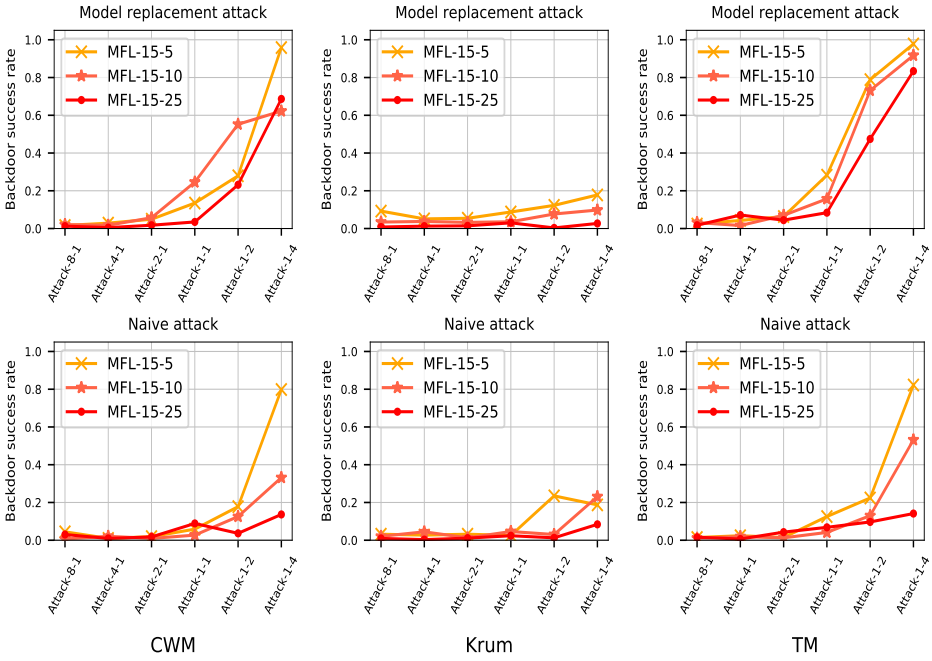
While Meta-FL enhances the robustness of all 6 methods, we observe that Krum benefits the most from our framework. We believe that lower variance on cohort aggregates aids Krum in effectively separating benign and malicious updates. We note that the server can further decrease the variance of cohort aggregates along each coordinate by increasing the size of training cohorts, As discussed in Section B.4, and improve the robustness of the Krum aggregation rule.

Moreover, other methods such as coordinate-wise median and trimmed mean which are anomaly detection-based defenses can also benefit from lower variations on cohort aggregate. Perhaps the most important principle in detecting outliers is defining the distribution of ordinary observations, which can be easier if observations exhibit low variations. Figure B.5 shows the results for experiments in which we evaluate the performance of Krum, CWM and TM across Meta-FL frameworks with increasingly larger training cohorts. For this experiment, we set the number of cohorts to 15 and vary the cohort size between 5, 10, and 15. As reflected in Figure B.5, increasing the size of training cohorts improves the robustness of these techniques across all scenarios, especially for scenarios in which the adversary appears more frequently with more sybils.

Although defenses such as RFA, differential privacy, and norm bounding appear to be robust against poisoning attacks [258, 260], our empirical evaluations show that they are not



(a) SVHN



(b) GTSRB

Figure B.5: Effect of size of training cohorts on the efficacy of CWM, Krum and TM against backdoor attacks.

effective against backdoor attacks, specifically model replacement attacks. In poisoning attacks, the adversarial sub-task, which is the misclassification of unmodified data samples (e.g. classifying certain images of digit 1 as digit 7), is in direct contradiction with the primary learning task. Therefore, poisoning updates (or aggregates) face direct opposition from submissions of benign clients, which makes it harder for the adversary to succeed. However, in the case of backdoor attacks, the adversary's goal for the model is to learn the causal relationship between the presence of an attacker's chosen trigger and certain model output which does not require the model to learn any knowledge contradicting the primary learning task. Therefore, backdoor attacks tend to be stealthier compared to poisoning attacks, and defenses that have shown resilience against poisoning attacks might fall short against backdoor attacks.

B.6 Conclusion

In this study, we show that it is in fact possible to defend against backdoor attacks without violating the privacy of participating clients. We propose Meta-FL, a new federated learning framework that not only protects the privacy of participants through the secure aggregation protocol but also facilitates defense against backdoor attacks. Our empirical evaluations demonstrate that state-of-the-art defenses tend to be more effective against backdoor attacks in Meta-FL compared to baseline FL while offering the same or better utility. Our results suggest that not only does Meta-FL protect the privacy of participants but also optimizes the robustness-utility trade-off better than the baseline setting.

Bibliography

[1] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018.

[2] Ilhami Torunoglu and Edoardo Charbon. Watermarking-based copyright protection of sequential functions. *IEEE Journal of Solid-State Circuits*, 35(3):434–440, 2000.

[3] Chen Xi. Scan chain based hardware security. In *University of Maryland (UMD) Thesis and Dissertations)*. UMD, 2018.

[4] Aijiao Cui, Gang Qu, and Yan Zhang. Ultra-low overhead dynamic watermarking on scan design for hard ip protection. *IEEE Transactions on Information Forensics and Security*, 10(11):2298–2313, 2015.

[5] Huiying Li, Emily Wenger, Shawn Shan, Ben Y Zhao, and Haitao Zheng. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*, 2019.

[6] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, pages 1–12, 2019.

[7] Xi Chen, Gang Qu, Aijiao Cui, and Carson Dunbar. Scan chain based ip fingerprint and identification. In *2017 18th International Symposium on Quality Electronic Design (ISQED)*, pages 264–270. IEEE, 2017.

[8] Rafal Baranowski, Michael A Kochte, and Hans-Joachim Wunderlich. Fine-grained access management in reconfigurable scan networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(6):937–946, 2015.

[9] Dennis RE Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi B Tahoori. Analysis of transient voltage fluctuations in fpgas. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 12–19. IEEE, 2016.

[10] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. An inside job: Remote power analysis attacks on fpgas. *IEEE Design & Test*, 38(3):58–66, 2021.

[11] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[12] Abhishek Kumar, Benjamin Finley, Tristan Braud, Sasu Tarkoma, and Pan Hui. Sketching an ai marketplace: Tech, economic, and regulatory aspects. *IEEE Access*, 9:13761–13774, 2021.

[13] Research and Markets. Global $8.81 billion semiconductor intellectual property (ip) market to 2026 - royalty segment is expected to have a lucrative growth, Jan 2021. URL `https://www.globenewswire.com/en/news-release/2021/01/18/2159975/28124/en/Global-8-81-Billion-Semiconductor-Intellectual-Property-IP-Market.html`.

[14] Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16482–16491, 2021.

[15] Bo Luo, Yu Li, Lingxiao Wei, and Qiang Xu. On functional test generation for deep neural network ips. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1010–1015. IEEE, 2019.

[16] M. Ribeiro, K. Grolinger, and M. A. M. Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902, Dec 2015. doi: 10.1109/ICMLA.2015.152.

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

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

[19] Donghua Wang, Rangding Wang, Li Dong, Diqun Yan, Xueyuan Zhang, and Yongkang Gong. Adversarial examples attack and countermeasure for speech recognition system: A survey. In *International Conference on Security and Privacy in Digital Economy*, pages 443–468. Springer, 2020.

[20] Nuno Martins, José Magalhães Cruz, Tiago Cruz, and Pedro Henriques Abreu. Adversarial machine learning applied to intrusion and malware scenarios: a systematic review. *IEEE Access*, 8:35403–35419, 2020.

[21] Sara Kaviani, Ki Jin Han, and Insoo Sohn. Adversarial attacks and defenses on ai in medical imaging informatics: A survey. *Expert Systems with Applications*, page 116815, 2022.

[22] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

[23] Dudi Nassi, Raz Ben-Netanel, Yuval Elovici, and Ben Nassi. Mobilbye: attacking adas with camera spoofing. *arXiv preprint arXiv:1906.09765*, 2019.

[24] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. *ACM SIGARCH Computer Architecture News*, 42(3):361–372, 2014.

[25] P. Qiu, D. Wang, Y. Lyu, and G. Qu. Voltjockey: Breaking sgx by software-controlled voltage-induced hardware faults. In *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6, 2019. doi: 10.1109/AsianHOST47458.2019.9006701.

[26] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012.

[27] Mathieu Dumont, Mathieu Lisart, and Philippe Maurine. Electromagnetic fault injection: how faults occur. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 9–16. IEEE, 2019.

[28] Ronald Rivest. The md5 message-digest algorithm. Technical report, 1992.

[29] Philip Koopman and Tridib Chakravarty. Cyclic redundancy code (crc) polynomial selection for embedded networks. In *International Conference on Dependable Systems and Networks, 2004*, pages 145–154. IEEE, 2004.

[30] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.

[31] Neil F Johnson, Zoran Duric, and Sushil Jajodia. *Information hiding: steganography and watermarking-attacks and countermeasures: steganography and watermarking: attacks and countermeasures*, volume 1. Springer Science & Business Media, 2001.

[32] Husrev T Sencar and Nasir Memon. Watermarking and ownership problem: a revisit. In *Proceedings of the 5th ACM workshop on Digital rights management*, pages 93–101, 2005.

[33] Andrew B Kahng, John Lach, William H Mangione-Smith, Stefanus Mantik, Igor L Markov, Miodrag Potkonjak, Paul Tucker, Huijuan Wang, and Gregory Wolfe. Watermarking techniques for intellectual property protection. In *Proceedings of the 35th annual Design Automation Conference*, pages 776–781, 1998.

[34] Andrew B Kahng, Stefanus Mantik, Igor L Markov, Miodrag Potkonjak, Paul Tucker, Huijuan Wang, and Gregory Wolfe. Robust ip watermarking methodologies for physical design. In *Proceedings of the 35th annual Design Automation Conference*, pages 782–787, 1998.

[35] N Nalla Anandakumar, M Sazadur Rahman, Mridha Md Mashahedur Rahman, Rasheed Kibria, Upoma Das, Farimah Farahmandi, Fahim Rahman, and Mark M Tehranipoor. Rethinking watermark: Providing proof of ip ownership in modern socs. *Cryptology ePrint Archive*, 2022.

[36] Wei Liang, SUN Xingming, RUAN Zhiqiang, LONG Jing, and WU Chengtao. A sequential circuit-based ip watermarking algorithm for multiple scan chains in design-for-test. *Radioengineering*, 20(2), 2011.

[37] Yu-Cheng Fan. Testing-based watermarking techniques for intellectual-property identification in soc design. *IEEE Transactions on Instrumentation and Measurement*, 57(3):467–479, 2008.

[38] Amr T Abdel-Hamid, Sofiene Tahar, and El Mostapha Aboulhamid. Finite state machine ip watermarking: A tutorial. In *First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, pages 457–464. IEEE, 2006.

[39] Aijiao Cui, Chip-Hong Chang, Sofiene Tahar, and Amr T Abdel-Hamid. A robust fsm watermarking scheme for ip protection of sequential circuit design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(5):678–690, 2011.

[40] Arlindo L Oliveira. Techniques for the creation of digital watermarks in sequential circuit designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(9):1101–1117, 2001.

[41] Jorge M Pena and Arlindo L Oliveira. A new algorithm for exact reduction of incompletely specified finite state machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(11):1619–1632, 1999.

[42] Matthew Lewandowski, Richard Meana, Matthew Morrison, and Srinivas Katkoori. A novel method for watermarking sequential circuits. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 21–24. IEEE, 2012.

[43] Li Zhang and Chip-Hong Chang. State encoding watermarking for field authentication of sequential circuit intellectual property. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3013–3016. IEEE, 2012.

[44] Darko Kirovski and Miodrag Potkonjak. Intellectual property protection using watermarking partial scan chains for sequential logic test generation. In *Proc. IEEE High Level Design, Verification, and Test Conf.* Citeseer, 1998.

[45] Aijiao Cui and Chip-Hong Chang. Intellectual property authentication by watermarking scan chain in design-for-testability flow. In *2008 IEEE International Symposium on Circuits and Systems*, pages 2645–2648. IEEE, 2008.

[46] Aijiao Cui and Chip-Hong Chang. A post-processing scan-chain watermarking scheme for vlsi intellectual property protection. In *2012 IEEE Asia Pacific Conference on Circuits and Systems*, pages 412–415. IEEE, 2012.

[47] Aijiao Cui and Chip-Hong Chang. An improved publicly detectable watermarking scheme based on scan chain ordering. In *2009 IEEE International Symposium on Circuits and Systems*, pages 29–32. IEEE, 2009.

[48] Chip-Hong Chang and Aijiao Cui. Synthesis-for-testability watermarking for field authentication of vlsi intellectual property. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(7):1618–1630, 2010.

[49] Aijiao Cui, Chip-Hong Chang, and Li Zhang. A hybrid watermarking scheme for sequential functions. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2333–2336. IEEE, 2011.

[50] Debasri Saha and Susmita Sur-Kolay. A unified approach for ip protection across design phases in a packaged chip. In *2010 23rd International Conference on VLSI Design*, pages 105–110. IEEE, 2010.

[51] Aijiao Cui, Wei Liang, and Gang Qu. A low-overhead dynamic watermarking scheme on scan design for easy authentication. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 778–781. IEEE, 2014.

[52] Xiaonan Huang, Aijiao Cui, and Chip-Hong Chang. A new watermarking scheme on scan chain ordering for hard ip protection. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017. doi: 10.1109/ISCAS.2017.8050823.

[53] Yue Li, Hongxia Wang, and Mauro Barni. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193, 2021.

[54] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval - ICMR '17*, 2017.

[55] Tianhao Wang and Florian Kerschbaum. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. IEEE, 2019.

[56] Enzo Tartaglione, Marco Grangetto, Davide Cavagnino, and Marco Botta. Delving in the loss landscape to embed robust watermarks into neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1243–1250. IEEE, 2021.

[57] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750*, 2018.

[58] Tianhao Wang and Florian Kerschbaum. Robust and undetectable white-box watermarks for deep neural networks. *arXiv preprint arXiv:1910.14268*, 1(2), 2019.

[59] Jiangfeng Wang, Hanzhou Wu, Xinpeng Zhang, and Yuwei Yao. Watermarking in deep neural networks via error back-propagation. *Electronic Imaging*, 2020(4):22–1, 2020.

[60] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[61] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[62] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1615–1631, 2018.

[63] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[64] Jia Guo and Miodrag Potkonjak. Evolutionary trigger set generation for dnn black-box watermarking. *arXiv preprint arXiv:1906.04411*, 2019.

[65] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 228–240, 2019.

[66] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[67] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 126–137, 2019.

[68] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[69] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[70] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 14–25, 2021.

[71] Jingjing Zhao, Qingyue Hu, Gaoyang Liu, Xiaoqiang Ma, Fei Chen, and Mohammad Mehedi Hassan. Afa: Adversarial fingerprinting authentication for deep neural networks. *Computer Communications*, 150:488–497, 2020.

[72] Siyue Wang, Xiao Wang, Pin-Yu Chen, Pu Zhao, and Xue Lin. Characteristic examples: High-robustness, low-transferability fingerprinting of neural networks. In *IJCAI*, pages 575–582, 2021.

[73] Chuan Guo, Jared S Frank, and Kilian Q Weinberger. Low frequency adversarial perturbation. *arXiv preprint arXiv:1809.08758*, 2018.

[74] Yash Sharma, Gavin Weiguang Ding, and Marcus Brubaker. On the effectiveness of low frequency perturbations. *arXiv preprint arXiv:1903.00073*, 2019.

[75] Xiaoxuan Lou, Shangwei Guo, Tianwei Zhang, Yinqian Zhang, and Yang Liu. When nas meets watermarking: ownership verification of dnn models via cache side channels. *arXiv preprint arXiv:2102.03523*, 2021.

[76] Tian Wang, Xiaoxin Cui, Dunshan Yu, Omid Aramoon, Timothy Dunlap, Gang Qu, and Xiaole Cui. Polymorphic gate based ic watermarking techniques. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 90–96. IEEE, 2018.

[77] Adrian Stoica, Ricardo Zebulum, Didier Keymeulen, and Jason Lohn. On polymorphic circuits and their design using evolutionary algorithms. 2002.

[78] Adrian Stoica, Ricardo Zebulum, and Didier Keymeulen. Polymorphic electronics. In *International Conference on Evolvable Systems*, pages 291–302. Springer, 2001.

[79] Adrian Stoica, RS Zebulum, Xin Guo, Didier Keymeulen, MI Ferguson, and Vu Duong. Taking evolutionary circuit design from experimentation to implementation: Some useful techniques and a silicon demonstration. *IEE Proceedings-Computers and Digital Techniques*, 151(4):295–300, 2004.

[80] Lukas Sekanina, Richard Ruzicka, Zdenek Vasicek, Roman Prokop, and Lukas Fujcik. Repomo32-new reconfigurable polymorphic integrated circuit for adaptive hardware. In *2009 IEEE Workshop on Evolvable and Adaptive Hardware*, pages 39–46. IEEE, 2009.

[81] Lukas Sekanina, Lukas Starecek, Zbysek Gajda, and Zdenek Kotasek. Evolution of multifunctional combinational modules controlled by the power supply voltage. In *First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, pages 186–193. IEEE, 2006.

[82] Lukas Sekanina, Richard Ruzicka, and Zbysek Gajda. Polymorphic fir filters with backup mode enabling power savings. In *2009 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 43–50. IEEE, 2009.

[83] Richard Ruzicka, Lukas Sekanina, and Roman Prokop. Physical demonstration of polymorphic self-checking circuits. In *2008 14th IEEE International On-Line Testing Symposium*, pages 31–36. IEEE, 2008.

[84] Lukas Sekanina, Lukas Starecek, Zdenek Kotasek, and Zbysek Gajda. Polymorphic gates in design and test of digital circuits. *Int. J. Unconv. Comput.*, 4(2):125–142, 2008.

[85] Lukas Sekanina. Evolution of polymorphic self-checking circuits. In *International Conference on Evolvable Systems*, pages 186–197. Springer, 2007.

[86] Lukáš Sekanina. Evolutionary design of gate-level polymorphic digital circuits. In *Workshops on Applications of Evolutionary Computation*, pages 185–194. Springer, 2005.

[87] Julian F Miller, Dominic Job, and Vesselin K Vassilev. Principles in the evolutionary design of digital circuits—part i. *Genetic programming and evolvable machines*, 1(1): 7–35, 2000.

[88] Richard Ruzicka. On bifunctional polymorphic gates controlled by a special signal. *WSEAS Transactions on Circuits*, 7(3):96–101, 2008.

[89] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[90] Lukáš Sekanina. Design methods for polymorphic digital circuits. In *Proc. of the 8th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop DDECS*, pages 145–150, 2005.

[91] Gang Qu and Miodrag Potkonjak. *Intellectual property protection in VLSI designs: theory and practice*. Springer Science & Business Media, 2007.

[92] Gang Qu and Miodrag Potkonjak. Analysis of watermarking techniques for graph coloring problem. *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 190–193, 1998.

[93] Gang Qu and Miodrag Potkonjak. Intellectual property protection in vlsi designs: Theory and practice. *Kluwer Academic Publishers*, 2003.

[94] Omid Aramoon, Pin-Yu Chen, and Gang Qu. Don't forget to sign the gradients! *Proceedings of Machine Learning and Systems*, 3:194–207, 2021.

[95] Simon Du and Jason Lee. On the power of over-parametrization in neural networks with quadratic activation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1329–1338, 10–15 Jul 2018.

[96] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

[97] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[98] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred Hero, and Pramod K Varshney. A primer on zeroth-order optimization in signal processing and machine learning. *IEEE Signal Processing Magazine*, 2020.

[99] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. Zeroth-order stochastic variance reduction for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3727–3737, 2018.

[100] Sijia Liu, Pin-Yu Chen, Xiangyi Chen, and Mingyi Hong. signSGD via zeroth-order oracle. In *International Conference on Learning Representations*, 2019.

[101] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[102] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[103] Lior Wolf, Tal Hassner, and Itay Maoz. *Face recognition in unconstrained videos with matched background similarity*. IEEE, 2011.

[104] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[105] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1891–1898, 2014.

[106] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18, pages 133:1–133:8. ACM, 2018.

[107] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

[108] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902. IEEE, 2015.

[109] A. Tamhane and D. Dunlop. *Statistics and Data Analysis: From Elementary to Intermediate*. Prentice Hall, 2000.

[110] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[111] John Lach, William H Mangione-Smith, and Miodrag Potkonjak. Fpga fingerprinting techniques for protecting intellectual property. In *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference (Cat. No. 98CH36143)*, pages 299–302. IEEE, 1998.

[112] Tingyuan Nie, Yansheng Li, Lijian Zhou, and Masahiko Toyonaga. A multilevel fingerprinting method for fpga ip protection. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1789–1792. IEEE, 2013.

[113] Andrew E Caldwell, Hyun-Jin Choi, Andrew B Kahng, Stefanus Mantik, Miodrag Potkonjak, Gang Qu, and Jennifer L Wong. Effective iterative techniques for fingerprinting design ip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(2):208–215, 2004.

[114] Gang Qu and Miodrag Potkonjak. Fingerprinting intellectual property using constraint-addition. In *Proceedings of the 37th annual design automation conference*, pages 587–592, 2000.

[115] Dipanjan Roy and Anirban Sengupta. Low overhead symmetrical protection of reusable ip core using robust fingerprinting and watermarking during high level synthesis. *Future Generation Computer Systems*, 71:89–101, 2017.

[116] Tingyuan Nie, Lijian Zhou, and Zhe-Ming Lu. Fingerprinting methods for intellectual property protection using constraints in circuit partitioning. *IET Circuits, Devices & Systems*, 10(3):237–243, 2016.

[117] Carson Dunbar and Gang Qu. Satisfiability don't care condition based circuit fingerprinting techniques. In *The 20th Asia and South Pacific Design Automation Conference*, pages 815–820. IEEE, 2015.

[118] Carson Dunbar and Gang Qu. A practical circuit fingerprinting method utilizing observability don't care conditions. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

[119] Chip-Hong Chang and Li Zhang. A blind dynamic fingerprinting technique for sequential circuit intellectual property protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(1):76–89, 2013.

[120] Tian Wang, Xiaoxin Cui, Dunshan Yu, Omid Aramoon, Timothy Dunlap, Gang Qu, and Xiaole Cui. A novel polymorphic gate based circuit fingerprinting technique. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pages 141–146, 2018.

[121] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 137–143. IEEE, 2015.

[122] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan JV Rajendran, and Ozgur Sinanoglu. Sarlock: Sat attack resistant logic locking. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 236–241. IEEE, 2016.

[123] Bo Yang, Kaijie Wu, and Ramesh Karri. Secure scan: A design-for-test architecture for crypto chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2287–2293, 2006.

[124] Omid Aramoon, Xi Chen, and Gang Qu. A reconfigurable scan network based ic identification for embedded devices. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 479–484. IEEE, 2018.

[125] Rafal Baranowski, Michael A Kochte, and Hans-Joachim Wunderlich. Modeling, verification and pattern generation for reconfigurable scan networks. In *2012 IEEE International Test Conference*, pages 1–9. IEEE, 2012.

[126] IEEE Standards Association et al. Ieee standard for test access port and boundary-scan architecture. *IEEE Std*, 1149:1–444, 2013.

[127] Erik Larsson and Konstantin Sibin. Fault management in an ieee p1687 (ijtag) environment. In *DDECS*, page 7. Citeseer, 2012.

[128] Shantanu Gupta, Tarang Vaish, and Santanu Chattopadhyay. Flip-flop chaining architecture for power-efficient scan during test application. In *14th Asian Test Symposium (ATS'05)*, pages 410–413. IEEE, 2005.

[129] Erik Jan Marinissen, Vikram Iyengar, and Krishnendu Chakrabarty. A set of benchmarks for modular testing of socs. In *Proceedings. International Test Conference*, pages 519–528. IEEE, 2002.

[130] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[131] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[132] Sina Mohseni, Mandar Pitale, Vasu Singh, and Zhangyang Wang. Practical solutions for machine learning safety in autonomous vehicles. *arXiv preprint arXiv:1912.09630*, 2019.

[133] Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*, 2018.

[134] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[135] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

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

[137] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[138] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[139] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[140] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[141] Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, and Luigi Lo Iacono. All your clouds are belong to us: security analysis of cloud management interfaces. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 3–14, 2011.

[142] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[143] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Proflip: Targeted trojan attack with progressive bit flips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7718–7727, 2021.

[144] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13198–13207, 2020.

[145] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138. IEEE, 2017.

[146] Jiawang Bai, Baoyuan Wu, Yong Zhang, Yiming Li, Zhifeng Li, and Shu-Tao Xia. Targeted attack against deep neural networks via flipping limited weight bits. *arXiv preprint arXiv:2102.10496*, 2021.

[147] Kai Tian, Shuigeng Zhou, Jianping Fan, and Jihong Guan. Learning competitive and discriminative reconstructions for anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5167–5174, 2019.

[148] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

[149] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[150] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.

[151] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 481–490, 2019.

[152] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. *Advances in neural information processing systems*, 31, 2018.

[153] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[154] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[155] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

[156] Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. *Advances in Neural Information Processing Systems*, 34: 677–689, 2021.

[157] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020.

[158] Haoran Wang, Weitang Liu, Alex Bocchieri, and Yixuan Li. Can multi-label classification networks know what they don't know? *Advances in Neural Information Processing Systems*, 34:29074–29087, 2021.

[159] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[160] Igor M Quintanilha, Roberto de ME Filho, José Lezama, Mauricio Delbracio, and Leonardo O Nunes. Detecting out-of-distribution samples using low-order deep features statistics. 2018.

[161] Vahdat Abdelzad, Krzysztof Czarnecki, Rick Salay, Taylor Denounden, Sachin Vernekar, and Buu Phan. Detecting out-of-distribution inputs in deep neural networks using an early-layer output. *arXiv preprint arXiv:1910.10307*, 2019.

[162] Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with gram matrices. In *International Conference on Machine Learning*, pages 8491–8501. PMLR, 2020.

[163] Alireza Zaeemzadeh, Niccolo Bisagno, Zeno Sambugaro, Nicola Conci, Nazanin Rahnavard, and Mubarak Shah. Out-of-distribution detection using union of 1-dimensional subspaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9452–9461, 2021.

[164] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyxCxhRcY7`.

[165] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ryiAv2xAZ`.

[166] Jingyang Zhang, Nathan Inkawhich, Yiran Chen, and Hai Li. Fine-grained out-of-distribution detection with mixup outlier exposure. *arXiv preprint arXiv:2106.03917*, 2021.

[167] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[168] Ambuj Tewari and Peter L Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8(5), 2007.

[169] Sunil Thulasidasan, Sushil Thapa, Sayera Dhaubhadel, Gopinath Chennupati, Tanmoy Bhattacharya, and Jeff Bilmes. An effective baseline for robustness to distributional shift. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 278–285. IEEE, 2021.

[170] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[171] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[172] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[173] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NIPS 2017 Workshop on Machine Learning and Computer Security*, volume 1, page 8, 2017.

[174] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016.

[175] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

[176] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

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

[178] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1778–1787, 2018.

[179] Jiajun Lu, Theerasit Issaranon, and David Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE international conference on computer vision*, pages 446–454, 2017.

[180] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE international conference on computer vision*, pages 5764–5772, 2017.

[181] Fabio Carrara, Fabrizio Falchi, Roberto Caldelli, Giuseppe Amato, Roberta Fumarola, and Rudy Becarelli. Detecting adversarial example attacks to deep neural networks. In *Proceedings of the 15th international workshop on content-based multimedia indexing*, pages 1–7, 2017.

[182] Stefanos Pertigkiozoglou and Petros Maragos. Detecting adversarial examples in convolutional neural networks. *arXiv preprint arXiv:1812.03303*, 2018.

[183] Fabio Carrara, Rudy Becarelli, Roberto Caldelli, Fabrizio Falchi, and Giuseppe Amato. Adversarial examples detection in features distance spaces. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

[184] Zhihao Zheng and Pengyu Hong. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[185] Jonathan Aigrain and Marcin Detyniecki. Detecting adversarial examples and other misclassifications in neural networks by introspection. *arXiv preprint arXiv:1905.09186*, 2019.

[186] Shiqing Ma and Yingqi Liu. Nic: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*, 2019.

[187] Julia Lust and Alexandru Paul Condurache. Gran: an efficient gradient-norm based detector for adversarial and misclassified examples. *arXiv preprint arXiv:2004.09179*, 2020.

[188] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[189] Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, Wenchang Shi, and Xiaofeng Wang. Detecting adversarial image examples in deep neural networks with adaptive noise reduction. *IEEE Transactions on Dependable and Secure Computing*, 18(1):72–85, 2018.

[190] Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020.

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

[192] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *2015 IEEE international conference on data mining*, pages 301–309. IEEE, 2015.

[193] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

[194] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[195] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[196] Minghai Qin, Chao Sun, and Dejan Vucinic. Robustness of neural networks against storage media errors. *arXiv preprint arXiv:1709.06173*, 2017.

[197] Hui Guan, Lin Ning, Zhen Lin, Xipeng Shen, Huiyang Zhou, and Seung-Hwan Lim. In-place zero-space memory protection for cnn. *Advances in Neural Information Processing Systems*, 32, 2019.

[198] Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. Radar: Run-time adversarial weight attack detection and accuracy recovery. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 790–795. IEEE, 2021.

[199] Mojan Javaheripi and Farinaz Koushanfar. Hashtag: Hash signatures for online detection of fault-injection attacks on deep neural networks. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[200] Yu Li, Min Li, Bo Luo, Ye Tian, and Qiang Xu. Deepdyve: Dynamic verification for deep neural networks. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 101–112, 2020.

[201] Zecheng He, Tianwei Zhang, and Ruby Lee. Sensitive-sample fingerprinting of deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4729–4737, 2019.

[202] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[203] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1345–1362, 2020.

[204] Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. Towards security threats of deep learning systems: A survey. *IEEE Transactions on Software Engineering*, 2020.

[205] Omid Aramoon, Pin-Yu Chen, and Gang Qu. Aid: Attesting the integrity of deep neural networks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 19–24, 2021. doi: 10.1109/DAC18074.2021.9586290.

[206] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[207] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[208] Markus Kasper, Amir Moradi, Georg T Becker, Oliver Mischke, Tim Güneysu, Christof Paar, and Wayne Burleson. Side channels as building blocks. *Journal of Cryptographic Engineering*, 2(3):143–159, 2012.

[209] Mark Randolph and William Diehl. Power side-channel attack analysis: A review of 20 years of study for the layman. *Cryptography*, 4(2):15, 2020.

[210] Saurav Maji, Utsav Banerjee, and Anantha P Chandrakasan. Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels—attacks and defenses. *IEEE Internet of Things Journal*, 8(15):12079–12092, 2021.

[211] Yun Xiang, Zhuangzhi Chen, Zuohui Chen, Zebin Fang, Haiyang Hao, Jinyin Chen, Yi Liu, Zhefu Wu, Qi Xuan, and Xiaoniu Yang. Open dnn box by power side-channel attack. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(11):2717–2721, 2020.

[212] Vincent Meyers, Dennis Gnad, and Mehdi Tahoori. Reverse engineering neural network folding with remote fpga power analysis. In *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 1–10. IEEE, 2022.

[213] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. Deepem: Deep neural networks model recovery through em side-channel information leakage. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 209–218. IEEE, 2020.

[214] Kota Yoshida, Mitsuru Shiozaki, Shunsuke Okura, Takaya Kubota, and Takeshi Fujino. Model reverse-engineering attack against systolic-array-based dnn accelerator using correlation power analysis. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 104(1):152–161, 2021.

[215] Yun Xiang, Yongchao Xu, Yingjie Li, Wen Ma, Qi Xuan, and Yi Liu. Side-channel gray-box attack for dnns. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(1): 501–505, 2020.

[216] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, and Philippe Loubet-Moundi. High-speed ring oscillator based sensors for remote side-channel attacks on fpgas. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–8. IEEE, 2019.

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

[218] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 393–406, 2018.

[219] Yicheng Zhang, Rozhin Yasaei, Hao Chen, Zhou Li, and Mohammad Abdullah Al Faruque. Stealing neural network structure through remote fpga side-channel analysis. *IEEE Transactions on Information Forensics and Security*, 16:4377–4388, 2021.

[220] Shanquan Tian, Shayan Moini, Adam Wolnikowski, Daniel Holcomb, Russell Tessier, and Jakub Szefer. Remote power attacks on the versatile tensor accelerator in multi-tenant fpgas. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 242–246. IEEE, 2021.

[221] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium (WTS)*, pages 1–5. IEEE, 2018.

[222] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[223] github. Qonnx: Arbitrary-precision quantized neural networks in onnx, 2020. URL `https://github.com/fastmachinelearning/qonnx`.

[224] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 65–74, 2017.

[225] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[226] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[227] Y Bulatov. Notmnist dataset, 2020. URL `http://yaroslavvb.com/upload/notMNIST/`.

[228] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[229] Fangzhen Zhao, Chenyi Zhang, Naipeng Dong, Zefeng You, and Zhenxin Wu. A uniform framework for anomaly detection in deep neural networks. *Neural Processing Letters*, pages 1–22, 2022.

[230] Jonas Krautter, Dennis Gnad, and Mehdi Tahoori. Cpamap: on the complexity of secure fpga virtualization, multi-tenancy, and physical design. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 121–146, 2020.

[231] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1309–1326, 2020.

[232] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005.

[233] Smitha Milli, Ludwig Schmidt, Anca D Dragan, and Moritz Hardt. Model reconstruction from model explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 1–9, 2019.

[234] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[235] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[236] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. A framework for the extraction of deep neural networks by leveraging public data. *arXiv preprint arXiv:1905.09165*, 2019.

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

[238] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 105–113, 2019.

[239] Omid Aramoon, Pin-Yu Chen, Gang Qu, and Yuan Tian. Meta federated learning. *arXiv preprint arXiv:2102.05561*, 2021.

[240] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[241] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[242] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[243] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[244] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[245] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910*, 2018.

[246] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[247] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[248] Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv preprint arXiv:1808.10307*, 2018.

[249] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[250] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[251] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*, 2018.

[252] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2041–2055, 2019.

[253] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *arXiv preprint arXiv:1909.02742*, 2019.

[254] Gabriela F Cretu, Angelos Stavrou, Michael E Locasto, Salvatore J Stolfo, and Angelos D Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 81–95. IEEE, 2008.

[255] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[256] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[257] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

[258] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[259] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[260] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[261] Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

[262] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses. *arXiv preprint arXiv:1903.09860*, 2019.

[263] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[264] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkgyS0VFvr`.

[265] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[266] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[267] John A. Rice. *Mathematical Statistics and Data Analysis.* Belmont, CA: Duxbury Press., third edition, 2006.

[268] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.