Janet Elchynski

From:	Bill Suski
Sent:	Monday, January 25, 2021 9:50 AM
To:	aphipps@spawar.navy.mil; serge.leef@darpa.mil; scott.wenzel@darpa.mil; dcma.lee.hq.list.S2101A-casd@mail.mil; ResearchSupport@darpa.mil; ReportsDSO@darpa.mil
Cc:	Uy, Wes (contr-mto); Contracts
Subject:	Final Technical Report - BioSec: DaMIAN (AEC) - HR00112020057
Attachments:	AEC-BioSec-DaMIAN_FinalTechnicalReport_HR00112020057.pdf

All,

I have attached the Final Technical Report for the BioSec: DaMIAN effort (HR00112020057) on behalf of Applied Engineering Concepts, Incorporated (AEC). I will be sending an additional email to a subset of the above addressees (those listed to receive Special Technical Reports in the award document) containing additional deliverables.

Please let me know if there are any questions or anyone would like additional information. We appreciate the opportunity and look forward to potential future collaboration. Thanks!

V/r,

Bill

William Suski, Ph.D.
Principal Engineer
Applied Engineering Concepts, Incorporated
843-861-6758

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
01/25/2021	Final	08/26/2020 - 11/25/2020

4. TITLE AND SUBTITLE

Final Technical Report: DARPA BioSEC: Detection and Mitigation Intelligent Agent Network (DaMIAN)

5a. CONTRACT NUMBER

5b. GRANT NUMBER
HR00112020057

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Suski, William, Dr.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Applied Engineering Concepts, Incorporated
1904 Musgrave Ritual Drive
Eldersburg, MD 21784

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency (DARPA)
Contracts Management Office
675 North Randolph Street
Arlington, VA 22203-2114

10. SPONSOR/MONITOR'S ACRONYM(S)
DARPA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A - Approved for Public Release, Distribution is Unlimited
Per our Cooperative Agreement, "DARPA expects the work performed under this award to be fundamental research, and it is, therefore, not subject to publication restrictions"

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of the Detection and Mitigation Intelligent Agent Network (DaMIAN) effort is to develop human immune system (HIS)-inspired, antibody-like intelligent agents capable of detecting and mitigating hardware trojan attacks via the use of neuromorphic computational System-on-Chip (SoC) modules. Hardware trojan threats are not unlike those that the human body faces on a daily basis. The HIS is capable of detecting and eliminating a wide range of threats using both innate and adaptive defenses. Innate defenses begin operating as soon as one is born, patrol the body, and attack invaders. Adaptive defenses begin creating antibodies capable of eradicating invaders after an exposure or invasion. Therefore, the immune system can be framed as a multi-agent system (MAS) that includes an inherited understanding of itself as well as the capability to learn the difference between domestic entities and those that are foreign. The DaMIAN effort focused on baselining the SoC [Body] through monitoring of network-on-chip (NoC) transactions, detecting rare events [Invaders] in NoC traffic, and monitoring and overruling untrusted SoC modules through behavioral prediction, e.g. modeling or fingerprinting. Long term, this approach could result in SoC technology that is capable of continually operating in the face of a variety of attacks.

15. SUBJECT TERMS

hardware trojan detection, machine learning, long short-term memory, random network distillation, immune system

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Dr. William Suski
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	SAR	20	**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U			843-861-6758

Applied Engineering Concepts, Inc.

1904 Musgrave Ritual Drive
Eldersburg, MD 21784

Phone 410.215.1717
Fax 443.279.6238

DARPA BioSec:

Detection and Mitigation Intelligent Agent Network (DaMIAN)

Final Report

January 2021

Principal Investigator: Dr. William Suski
Principal Engineer
Applied Engineering Concepts, Incorporated

Contact Information: wsuski@ae-concepts.com
843-861-6758

Program Manager: Dr. Serge Leef, MTO

Cooperative Agreement #: HR00112020057

Period of Performance: 26 August 2020 - 25 November 2020

Contents

List of Acronyms

ACAP Adaptive Compute Acceleration Platform
AI artificial intelligence
ALU arithmetic logic unit
ASIC application-specific integrated circuit
CNN Convolutional Neural Network
CPI critical program information
CPU central processing unit
DaMIAN Detection and Mitigation Intelligent Agent Network
DARPA Defense Advanced Research Projects Agency
DL deep learning
DPU Deep Learning Processing Unit
DNN deep neural network
DOD Department of Defense
DSP digital signal processor
DSSoC Domain-Specific System on Chip
ECG electrocardiogram
FC Fully-Connected
FPGA field-programmable gate array
FPR false positive rate
GPU graphics processing unit
GRU Gated Recurrent Unit
HIS human immune system
IARPA Intelligence Advanced Research Projects Activity
IC integrated circuit
ID in-distribution
ILA Instruction-Level Abstraction
IIoT Industrial Internet of Things (IoT)
IoT Internet of Things
IP intellectual property
LSTM Long Short-Term Memory
MAS multi-agent system

ML machine learning
MSE mean squared error
NoC network-on-chip
NPE Neural Processing Engine
NPU Neural Network Processing Unit
NSF National Science Foundation
OOD out-of-distribution
RL reinforcement learning
RND Random Network Distillation
RNN Recurrent Neural Network
SDK software development kit
SFF small form factor
SoC System-on-Chip
SoM System-on-Module
SVM Support Vector Machine
SWaP size, weight, and power
TPR true positive rate
TPU Tensor Processing Unit

1 Introduction

The goal of the Detection and Mitigation Intelligent Agent Network (DaMIAN) effort is to develop human immune system (HIS)-inspired, antibody-like intelligent agents capable of detecting and mitigating hardware trojan attacks via the use of neuromorphic computational System-on-Chip (SoC) modules. This includes:

- Baselining the SoC [Body] through monitoring of network-on-chip (NoC) transactions,
- Detecting rare events [Invaders] in NoC traffic,
- Monitoring and Overruling [Surround and Attack] untrusted SoC modules through behavioral prediction, e.g. modeling or fingerprinting.

This report provides relevant background information, in Section 2, which attempts to link elements of the human immune system to the DaMIAN effort. It also outlines the progress made during execution of the three month effort including the Literature Review, Technical Opportunity Analysis, and Simulations which are detailed in Sections 3, 4, and 5 respectively.

2 Background

As previously mentioned, the goal of the DaMIAN effort is to explore the potential for developing hardware trojan detection and mitigation techniques that are inspired by the HIS. Long term, this approach has the potential to develop SoC technology that is capable of continually operating in the face of a variety of attacks. As indicated in [1], the potential for hardware trojan attacks on integrated circuits (ICs), including SoC technology, has become a significant concern. These attacks are typically deployed in untrusted foundries or design houses that add/modify hardware designs to include malicious capabilities. Because field-programmable gate array (FPGA) hardware trojans have already been found in deployed hardware, these types of attacks should no longer be considered potential threats [2].

2.1 Immune System Inspiration

Hardware trojan threats are not unlike those that the human body faces on a daily basis. The HIS is capable of detecting and eliminating a wide range of threats using both innate and adaptive defenses. Innate defenses begin operating as soon as one is born, patrol the body, and attack invaders [3]. These defenses are inherited and active from birth. On the other hand, adaptive defenses begin creating antibodies capable of eradicating invaders after an exposure or invasion. These antibodies are a learned defense and act as a type of memory for the HIS because they remain in the system for a period of time [4].

Therefore, the immune system can be framed as a multi-agent system (MAS) that includes an inherited understanding of itself as well as the capability to learn the difference between domestic (internal or "self") entities and those that are foreign (external or "non-self"), i.e. a baseline for the body. These agents can be pre-trained with innate defenses based on data from pre- and post-silicon simulation, verification, and validation. Furthermore, the multi-agent system (MAS) literature based on reinforcement learning has a concept called experience replay [5] that is analogous to the memory that antibodies provide for the immune system. The approach to hardware

trojan detection and mitigation pursued in this work assumes that attacks will occur and intends to appropriately instrument a SoC to handle an invasion. The rest of this report provides details of the author's early exploration into the use of antibody-like intelligent agents to detect and mitigate hardware trojan attacks through a combination of deep learning, SoC building blocks, and module fingerprinting [6].

2.2 Technology

It is worth noting that the full complement of technology required to implement deep learning (DL)-based, HIS-inspired, SoC trojan detection techniques does not currently exist. Today, neuromorphic computation is dominated by large discrete components like graphics processing units (GPUs) and application-specific integrated circuits (ASICs) [7]. Very recently, FPGAs have become available with embedded artificial intelligence (AI) modules [8] and true AI engines will be arriving in the near term (1-2 years) [9]. As technology continues to evolve, DL SoC modules will inevitably be developed and used as building blocks, such as those envisioned by the Defense Advanced Research Projects Agency (DARPA) Domain-Specific System on Chip (DSSoC) program [10]. In the next decade or so, there is the potential for smaller and significantly faster neuromorphic photonic processors [11] capable of light-based neural network processing. The DaMIAN effort intends to explore the potential for embedding DL modules within SoCs to behave as antibody-like agents, detecting rare events, and mitigating attacks through black-box modeling. These artificial antibodies would be positioned to monitor transactions at various locations within an SoC and are likely to be particularly effective in monitoring NoC activity as this type of architecture becomes more prevalent within SoC designs. Section 4 provides additional details on current and future technology that will enable HIS-inspired DL approaches to SoC security.

2.3 Larger Context

The BioSec DaMIAN effort is one of many on-going programs within the U.S. Government's portfolio focused on microelectronics security, including DARPA, IARPA, and NSF. Table 1 briefly describes a few of these efforts to place DaMIAN within this larger context. Each of these programs has a different set of goals and addresses different security threats. For example, the SHIELD program is focused on mitigating the threat of the introduction of fraudulent products into the microelectronics supply chain, which also indirectly effects critical program information (CPI) theft. POSH aims to create an open source hardware ecosystem based on mathematically provable secure electronics modules, i.e. intellectual property (IP) blocks, and validation tools that create a foundation for the rapid development of secure and complex SoCs. In a similar block-focused manner, the DSSoC program is focused on improving domain-specific performance of SoCs through the development and integration of heterogeneous processing components. DaMIAN can be considered as a security-focused application complement to these two programs. The secure, performance-enhanced SoC components that are developed within these two programs will enable the type of underlying hardware needed to execute the DL detection and mitigation techniques being explored in the DaMIAN effort in the form factor needed to protect SoCs. Additionally, techniques like DaMIAN may lead to a more general framework for developing SoC-specific trojan detection and mitigation techniques without requiring a significant manual effort through its data-driven approach and appropriate use of abstraction, such as Instruction-Level Abstraction (ILA).

Program	Description	Sponsor
DaMIAN	Monitor untrusted IP and other components to detect and mitigate trojan attacks	DARPA
DSSoC	Develop a SoC comprised of many cores that mix general purpose processors, special purpose processors, hardware accelerators, memory, and input/output devices to significantly improve performance of applications within a domain	DARPA
AISS	Create a novel, automated chip design flow that will allow security mechanisms to scale consistently with the goals of a chip design	DARPA
IDEA	Develop a general purpose hardware compiler for no-human-in-the-loop translation of source code or schematic to physical layout for SoCs	DARPA
POSH	Enable mathematically provable secure electronics and create an open source hardware IP ecosystem, along with accompanying validation tools	DARPA
RTML	Building an end-to-end general purpose compiler that can transform a high level ML framework into Verilog	DARPA
SHIELD	Verify the authenticity of components at every point in the supply chain	DARPA
STARSS	Design & manufacture chips and systems that are secure, trustworthy, assured, and resistant to attack or counterfeit	NSF
TIC	DisaggregateASICs into non-functional parts	IARPA

Table 1: Various U.S. Government microelectronics programs.

3 Literature Review

In [12], a heavily used taxonomy of hardware trojans is presented. This work breaks hardware trojans into three high-level categories based on physical, activation, and action characteristics. Physical characteristics include the distribution, structure, size, and type of trojan circuit that is implanted. At the top level, activation or trigger characteristics are divided into internal and external activation. Action or payload characteristics are broken down based on the intended trojan effect, e.g. transmit information, modify specification, and/or modify function. Based on discussions with Dr. Leef, the focus of the DaMIAN effort is detection of hardware trojans through detection of the trigger "signal" or activation mechanism. Detection of the trojan payload is considered to be a less tractable problem and also occurs after the fact.

A survey on the use of machine learning to detect and mitigate hardware trojan attacks is provided in [13]. One interesting piece of the paper is a table that breaks down the main innovations of machine learning (ML) within the field of hardware trojan detection. They highlight publications that utilize ML techniques for circuit reverse engineering, feature analysis, and side-channel analysis. However, they do not mention the use of ML techniques for trigger detection, which is the focus of the DaMIAN project.

One main assumption underlying the DaMIAN effort is that hardware trojans, or more specifically their trigger signals, are rare. If they were not rare, it would be much easier to discover triggers or activations during design verification and/or pre/post-silicon testing efforts, especially regression testing. The trigger could activate a set of wires or logic within an IP module that are rarely utilized, as considered in [14] and [15], or when a specific set of inputs is provided as described in a number of the benchmark IP cores on trust-hub.org [16, 17], e.g. input text plain of $0x44444444$. The latter activation method is the focus of the DaMIAN effort where input data is baselined to determine what is normal and then used to detect abnormal inputs.

3.1 Anomaly Detection

During the course of this literature review two relevant works that bring together DL and anomaly or rare event detection were identified. As described previously, the author assumes that trojan trigger events are rare. In [18], the authors present a method to incentivize reinforcement learning agents to enter new or novel states. This is done by training a predictor neural network to match the output of a fixed, randomly initialized target neural network. The prediction error is shown to be larger than expected when the network input values, or current observation, were not part of the original training dataset. The authors are able to generate a bonus reward based on this prediction error that results in the training of an intelligent agent that excels at exploration-based Atari games, e.g. Montezuma's Revenge.

The second work demonstrates anomaly detection using an Long Short-Term Memory (LSTM) neural network [19]. In this work, an LSTM network is trained to predict normal behavior and then uses the distribution of the prediction error to identify abnormal behaviors based on the approach detailed in [20]. The authors indicate that "[t]his is particularly helpful in real-world anomaly detection scenarios where instances of normal behaviour may be available in abundance but instances of anomalous behaviour are rare" which is expected to be the case for hardware trojans. A stacked LSTM network is applied to anomaly detection in electrocardiogram (ECG), space shuttle, power demand, and engine time series data. This type of networks consists of multiple layers

of cells as shown in Figure 1. The LSTM network in this work was trained on baseline data, i.e. non-anomalous, to learn to predict future data values. The prediction errors, based on multiple predictions for each discrete time, are then modeled as a multi-variate Gaussian distributions and the distribution parameters are estimated using the LSTM output. These parameters allow likelihood estimates to be calculated. Based on these estimates, a likelihood threshold is selected to detect anomalies. A similar approach is taken in the simulations described in Section 5, but an Support Vector Machine (SVM) [21] is used to detect anomalies instead of a likelihood threshold. More details on the LSTM are provided in Section 3.2.

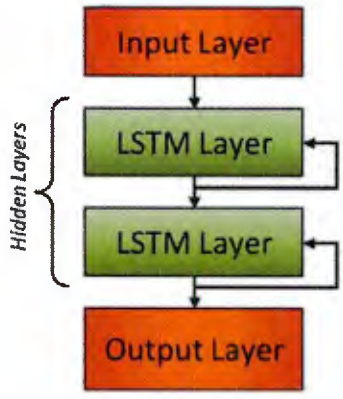


Figure 1: Stacked LSTM network, adapted from [19].

Process monitoring using time series analysis, for the purposes of fault detection, safety, quality control, etc., has been a topic of research for many years [22]. This research has traditionally focused on using cumulative sums and exponentially weighted moving averages to detect abrupt changes, i.e. anomalies, over pre-determined time windows. More recently, LSTM networks [23] have shown the ability to learn long-term correlations without requiring pre-specified window lengths. They also avoid the "vanishing gradient" problem that affects other Recurrent Neural Networks (RNNs) architectures. These advantages have led to wide ranging application of LSTM networks in fields such as connected handwriting recognition [24], speech processing [25], and translation [26].

As alluded to at the end of Section 2.3, the concept of abstraction is important to DaMIAN's ability to automatically generalize to a wide range of SoCs. One abstraction that could provide the appropriate generalization of the data in the DL training process is ILA [27]. It is expected that an ILA or ILA-like representation of SoC cores, specifically inputs, could be used to develop a general process of generating training/testing data sets, pre- and post-silicon, based on ILA representations of SoC inputs. This generalization comes from the fact that ILA attempts to represent all manner of SoCs as accelerators, e.g. central processing unit (CPU), GPU, etc., and builds a requisite instruction set to represent the various inputs/outputs. This is especially relevant as our approach is scaled to more complex IP cores, e.g. CPU or accelerator cores. There are also a number of tools that have been developed to simulate NoC traffic [28, 29, 30] which could provide a starting point for large scale, NoC-based simulations.

3.2 Long Short-Term Memory Network

An LSTM network is a type of RNN. RNNs are neural networks that contain loops to allow for information to persist within and amongst individual cells. This makes them well-suited to working with sequences of data. However, it has been shown to be difficult to use RNNs to connect information across long time delays [31]. LSTMs networks do not have this problem and were specifically designed to avoid it.

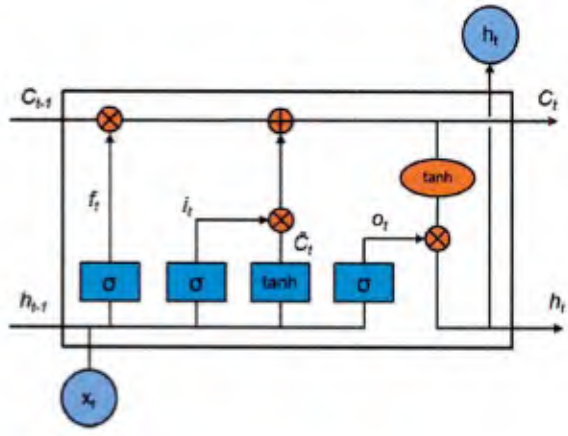


Figure 2: LSTM cell, adapted from [32].

LSTMs are constructed as a chain of repeated cells. Figure 2 shows the structure of a single LSTM cell. It is made up of four neural network layers: three sigmoid (σ) gates, and one inverse tangent ($tanh$) which are all shown in teal squares. There are also five point-wise operations: three multiplication, two addition, and another inverse tangent (shown in the orange ellipses). The black lines are vector connections which are concatenated when joining at nodes and copied when leaving or diverging from nodes. The three sigmoid gates decide how much of each component is allowed to affect the cell state, C_{t-1}, which passes across the top of the cell. The first is the "forget" gate layer, f_t, which decides how much of the previous cell state to discard based on the previous output, h_{t-1}, and the current input, x_t. The second is the "input" gate layer, i_t, which decides which values of the cell state will be updated. This works in conjunction with the inverse tangent layer which creates a vector of potential new state values, $\tilde{C}_t$. The cell state is updated by first point-wise multiplying with f_t to forget information and then by adding $i_t * \tilde{C}_t$ to produce the new state values. The "output" gate layer, o_t, decides which elements of the new state, C_t, will be provided as output, h_t. To do this, the state is first scaled to fit the range $(-1, 1)$ by the $tanh$ and then the outputs are multiplied by the σ gate layer. This is a basic LSTM cell. Many other variations have been developed including the Gated Recurrent Unit (GRU) which has seen widespread application as well.

4 Technical Opportunity Analysis

Executing neuromorphic computational tasks requires significant processing capability which means large size, weight, and power (SWaP). However, SoCs are typically designed to meet the most strin-

gent SWaP requirements in small form factor (SFF) embedded systems. Therefore, the overhead required for security monitoring must be kept to a minimum.

The neural network architecture chosen to solve a problem affects the overall computational requirements of the solution and in turn the SWaP. Three of the most commonly used neural network architectures are Dense or Fully-Connected, Convolutional Neural Network (CNN), and LSTM. Each of these architectures requires a different number of parameters which typically drives both memory and processing requirements. Fully-Connected (FC) networks have been shown to require the most parameters and CNN the fewest with LSTM falling in the middle [33]. One significant advantage of the LSTM architecture is that it does not require additional parameters to support longer "lookback" times, i.e. long input sequences do not increase the overall size of the network. However, they will increase training time. This is not a significant issue when focused on inferencing tasks using pre-trained models. In this scenario, models are trained with larger, more computationally capable hardware and then optimized and deployed to embedded systems, e.g. SoCs, for online monitoring. It is expected that, in realistic baseline development scenarios for SoC IP cores, the ability to learn what is "normal" over long periods of time will be significant. Therefore, the LSTM architecture is expected to have significant practical benefits for the BioSec application beyond its ability to perform time series analysis and prediction.

Advancements in deep learning for sensor data processing have created a desire to push neural networks away from large cloud infrastructure and closer to the sensor data at the network edge [34]. Additionally, many of the latest applications of deep learning have strict latency requirements, e.g. autonomous driving, which cannot be met by remote cloud implementations. This is also a large component of the expected explosion of IoT and Industrial IoT (IIoT) devices. These factors have driven the development of ever smaller, lower power devices capable of performing neuromorphic computations.

There are currently four main categories of neuromorphic processor architectures on the market: GPUs, FPGAs, ASICs, and SoC. GPUs are the most flexible, from a programming standpoint, and provides more form factor options. On the large end, discrete GPU devices such as the RTX series based on NVIDIA's Turing architecture are widely used in cloud server and desktop configurations [35]. At the smaller end, the NVIDIA Jetson Xavier and Nano series are System-on-Module (SoM) implementations for embedded systems [36]. There is also the Qualcomm Snapdragon 8 series which includes their Neural Processing Engine (NPE) software development kit (SDK) which includes a GPU as well as a digital signal processor (DSP) that has a neural network-focused software abstraction layer [37]. Despite the small size and advanced capability, GPU-based solutions are currently much larger than can be considered for SoC implementations.

FPGA-based neuromorphic processing is a more recent development and is predominantly focused on AI inferencing much like the embedded GPUs from NVIDIA. Xilinx (recently acquired by AMD) and Intel are two of the biggest players in this space with their Versal [9] and Arria/Stratix 10 [38] devices, respectively. The Xilinx Versal is called an Adaptive Compute Acceleration Platform (ACAP) to indicate the integration of traditional FPGA fabric with software programmable processors and accelerator engines. Intel's (formerly Altera's) Arria is a more traditional FPGA with an AI-focused development tool suite and the Stratix 10 NX contains a new kind of embedded block called an AI Tensor Block that is optimized for matrix-matrix and vector-matrix calculations [39]. These devices, especially those with AI-specific hardware blocks, are the beginnings

of the type of hardware solution needed to enable SoC embedded AI cores which can be used for hardware trojan detection and mitigation.

In the ASIC-focused AI acceleration space, there are a number of high-profile competitors including Huawei and Google. Google's Cloud and Edge Tensor Processing Units (TPUs) are dedicated ASICs that complement CPUs, GPUs, and FPGAs to enable AI in the cloud and at the edge. This device is similar to the Ascend 310 [40] from HiSilicon, a subsidiary of Huawei. The Ascend 310 is an ASIC device focused on AI applications that is based on Huawei's Da Vinci Neural Network Processing Unit (NPU) architecture. These devices are leading the way towards SoC implementations of AI-specific hardware components. It is worth noting that certain equipment from Huawei, and its subsidiaries, is prohibited by Federal Acquisition Regulations based on section 889(a)(1)(B) of the John S. McCain National Defense Authorization Act (NDAA) for Fiscal Year (FY) 2019 [41].

The most relevant technologies to the DaMIAN effort are the SoC-based AI IP cores. These SoCs have been introduced quite recently and include offerings from HiSilicon and Xilinx. The HiSilicon Kirin 970 is a communications-focused SoC, also based on Huawei's Da Vinci NPUs architecture, that provides AI acceleration [42]. This technology is of particular relevance because it comprises both large and small NPUs for heavy and lightweight computing applications [43]. The Xilinx Zynq 7000 SoC contains a similar Deep Learning Processing Unit (DPU) core which is a configurable computational engine for CNNs [44]. These appear to be the first iterations of AI-specific accelerators that are available as IP cores for SoC integration. As more SoCs begin to include this type of computational component there will be more opportunities to include deep learning-based hardware trojan detection and mitigation techniques within SoCs.

5 Simulations

The simulation portion of this effort was focused on developing techniques capable of building a baseline of a simple IP core and using this baseline to identify rare events or anomalous behaviors. More specifically, simulations were developed to detect hardware trojan triggers using machine learning. To our knowledge, there has not yet been any work published in the manner described. Two different detection techniques were explored. The first is based on Random Network Distillation (RND) described in [18]. The second is based on the use of an LSTM network similar to what is described in [19].

5.1 Random Network Distillation Trigger Detectiion

The RND technique was developed to improve the performance of reinforcement learning (RL) algorithms playing Atari games, e.g Montezuma's Revenge, by rewarding the agent for finding new states [18].

Figure 3 depicts a simple approach to detecting novel input states based on RND. In this notional scenario, NoC data is provided as input to two deep neural networks (DNNs). One of these networks, the target, is fixed and contains randomly initialized weights for each node. The second network, the predictor, is trained with labeled input data to achieve output values that minimize the squared error between its own outputs and the target network based on the same input. Figure 4, from [18], depicts the test mean squared error (MSE) between predictor and target networks when samples from class "0" and varying proportions of a target class from the MNIST images are used

Figure 3: Overview of RND.

for training. As one would expect, the test MSE, calculated from held out target class samples, decreases as the number of samples from the target class used for training is increased.

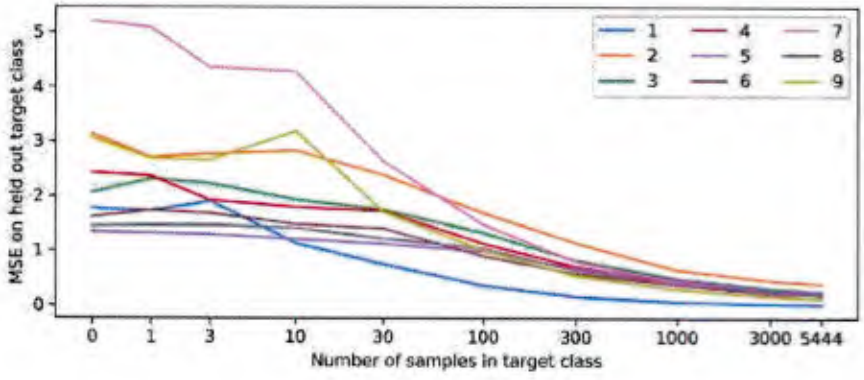


Figure 4: The MSE between the outputs of randomly initialized target network and trained predictor network provided the same inputs. The number of samples of a specific class (shown in the legend) used for training was limited to the values shown on the x axis [18].

As an initial introduction to RND, the previously described results are reproduced using randomly selected inputs to a simple arithmetic logic unit (ALU), depicted in Figure 5, instead of MNIST images. In this simulation, two sets of 100,000 random integer operands and varying proportions of two different opcodes, 0 for addition and 1 for subtraction, are provided as input to the predictor and target networks. The MSE, shown in Figure 6, also decreases as the proportion of samples from the target class is increased from 0% to 50%.

Using target and predictor networks trained with simulated ALU inputs, squared error values generated by the two networks can be compared for input data that is in-distribution (ID), i.e. valid input data [0, 256), [0, 256), and [0, 1], and out-of-distribution (OOD), i.e. [0, 256), [0, 256), [2, 256). This simulates an ALU that contains an undocumented opcode acts as trojan trigger. Figure 7a is a histogram of squared error values for 100,000 ID inputs. It can be seen that error values are clustered towards 0 and occurrences taper off quickly as squared error increases.

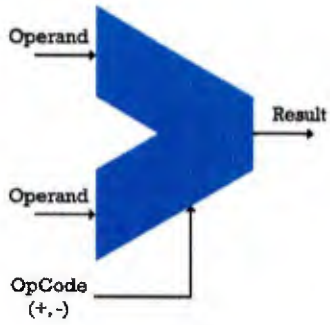


Figure 5: Simple ALU used for initial simulations.

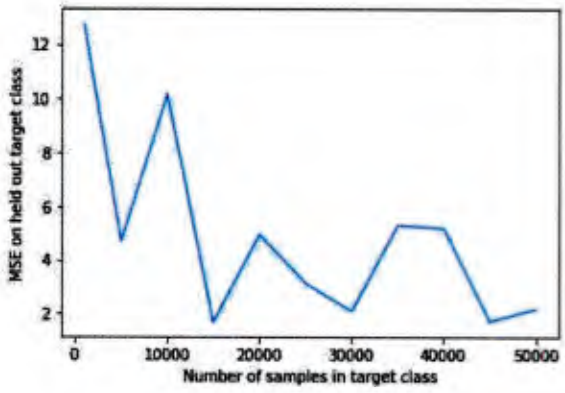
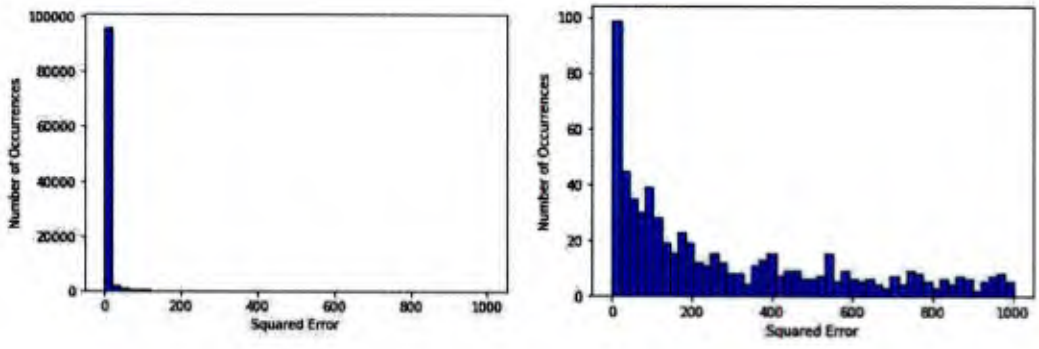


Figure 6: MSE between target and predictor networks resulting from random ALU inputs with varying proportions of opcode inputs.



(a) ID. (b) OOD.

Figure 7: Histogram of squared error values for ALU input values to predictor and target networks.

Figure 7b is a histogram of squared error values for 100,000 OOD inputs, i.e. invalid opcodes. It can be seen that the number of occurrences is low across the board. It is also worth noting that the scale of the y axis is almost two orders of magnitude smaller than shown in Figure 7a. Figure 10 is included to highlight the magnitude of the difference. This is an initial indication that RND could be used to detect rare/novel input states.

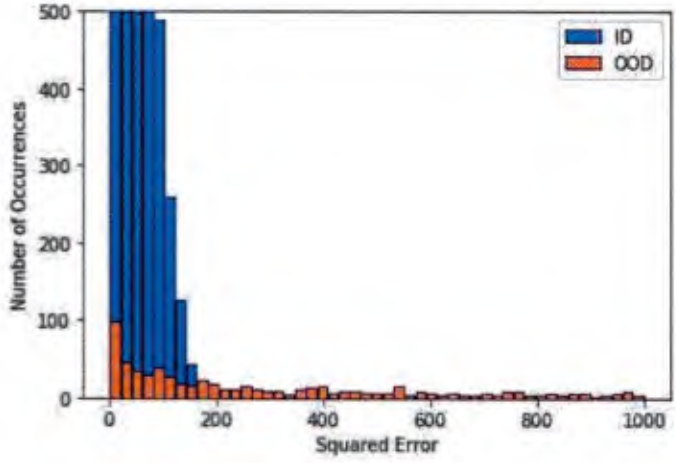


Figure 8: Histogram of squared error values for ID and OOD ALU input values to predictor and target networks.

Next, a SVM [21, 45] was trained to classify the squared error between the predictor/target network outputs for the two-opcode simulation described previously. Figure 9 is a confusion matrix for an instance of this end-to-end ID / OOD classification process which shows a distinct ability to detect opcodes that have not previously been experienced by the ALU. The true positive rate (TPR) for this experiment is 1.0 and the false positive rate (FPR) is 0.013.

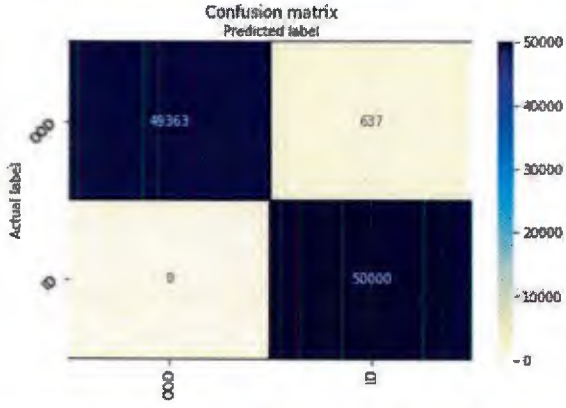


Figure 9: Confusion matrix for SVM classification of two-opcode squared error values.

For the next experiments, the initial two-opcode ALU was expanded into a four-opcode version

and similar tasks were performed. Figure 10 is again a histogram of the squared errors for ID and OOD ALU input values to predictor and target networks. It can be seen that there is still a significantly larger error (roughly one order of magnitude) between the ID and OOD squared error values. However, it is not as significant of a difference as was seen for the two-opcode version. This result is expected because intuition would suggest that the problem should get harder as the number of functions increases, i.e. the number of options.

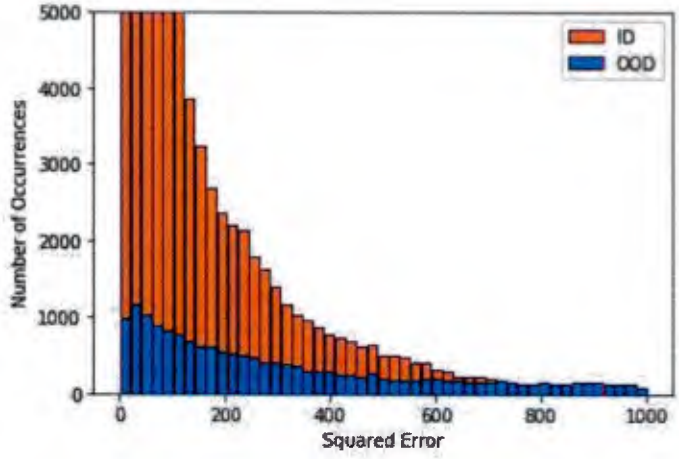


Figure 10: Histogram of squared error values for ID and OOD ALU input values to predictor and target networks.

Additionally, an SVM was trained to distinguish between ID and OOD based on the squared error values only. Figure 11 shows the corresponding confusion matrix for this classifier. Based on this confusion matrix, the TPR is found to be 1.0 and the FPR is 0.13. These two experiments demonstrate, in a limited setting, a potential undocumented functionality trigger of a hardware trojan circuit based solely on system inputs. It is anticipated that more complex neural networks and/or different architectures would be required as the complexity of the IP core of interest is increased. There is also the issue of other trigger types, such as sequentially triggered trojans, which is discussed further below.

In an attempt to detect sequential triggers based on the previously reported RND work, the number of inputs to the predictor and target networks was expanded to accept a set of N ALU inputs that simulate an ordered sequence of calculations. The predictor network was trained with a set of 100,000 uniformly distributed, randomly generated operands and opcodes for sequences of length $L = 5$, 10, and 15. This was considered to be the normal or ID dataset. The OOD dataset was again a set of 100,000 uniformly distributed, randomly generated operands but with repeated opcodes, i.e. $[0, 0, ..., 0]$ of length L. This simulation did not show any discernible difference in the squared error between the ID and OOD distribution data. This is thought to be due to the presence of repeated opcode sequences within the ID dataset that are quite similar to the trigger which does not result in significant differences in the squared error between the output and predictor networks. However, there is the potential that larger or more complex neural networks, including those with more outputs and different activations, would yield more discrimination power. This includes the

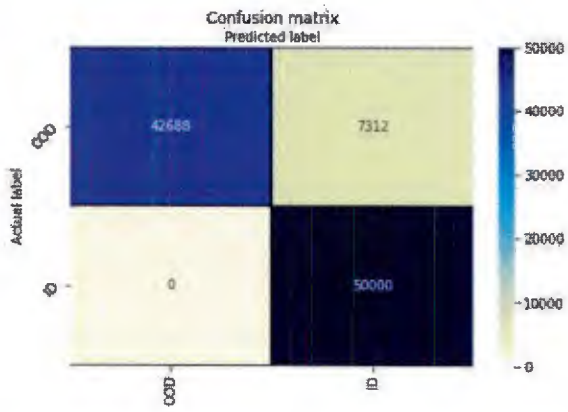


Figure 11: Confusion matrix for SVM classification of squared error values.

LSTM network architecture described in Section 3.

5.2 Long Short-Term Memory Trigger Detection

The next simulations undertaken were intended as an exploration of the use of the LSTM neural network architecture to detect sequential trojan triggers in a simulated ALU. The simulation is an adaptation of the method described for anomaly detection in [19] and detailed in Section 3. Figure 12 shows a block diagram of the overall anomaly detection process. The inputs to the process are the opcode at time t, o_t, along with the previous L opcodes. In this scenario, an opcode is simply an integer from the set $\{0, 1, 2, 3\}$.

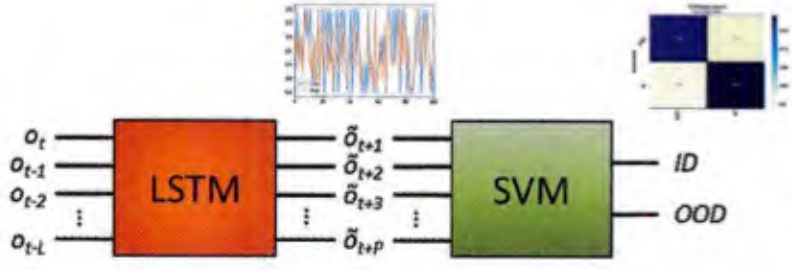


Figure 12: Block diagram of the LSTM-based anomaly detection method.

In an attempt to simulate a simple ALU's operational baseline, a dataset of $10000 + L$ opcodes is assembled by randomly selecting from a set of $S = 20$ randomly generated opcode sequences of length $N = 10$. This is considered to be the ID dataset. This data is scaled to lie in the range $r_{min} = 0$ to $r_{max} = 1$ using the sklearn MinMaxScaler [46] according to 2 where x is the set of input values and x_{scaled} is the output. After scaling, the opcode dataset is split into testing and training data with 67% of the data reserved for training and 33% for testing.

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

$$x_{scaled} = x_{std} * (r_{max} - r_{min}) + r_{min} \tag{2}$$

An LSTM network is trained for 10 epochs using this dataset using the Keras library [47]. These types of networks generally take inputs of shape (samples, timesteps, features). For this simulation, the LSTM network had input training data of shape (6700, $L + 1$, 1) with the "look back" value $L = 9$. Two hidden layers of LSTM cells are used with 50 cells in each layer. The outputs are a set of intermediate values, $\tilde{o}_{t+1}$ through $\tilde{o}_{t+P}$, which are $P = 10$ predictions of future opcodes.

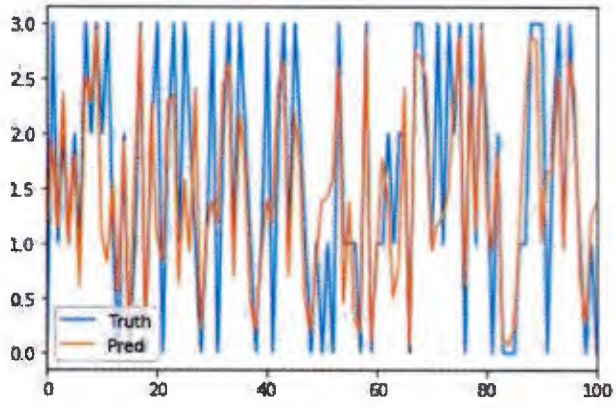


Figure 13: Opcode predictions for time $t + 1$ for 100 inputs to a trained LSTM network.

Figure 13 shows the opcode predicted for time $t + 1$ for 100 of the 3300 ID opcodes that were reserved for testing. This shows that this relatively small network was capable of quickly learning the various opcode sequences, i.e. the ALU baseline. It is expected that a real world IP core would provide a much noisier set of inputs containing sequences of behaviors of unknown length.

Figure 14 is a plot of the mean squared error and the mean error standard deviation for the 3300 test opcode inputs predicted into the future from $t + 1$ to $t + 10$. It matches intuition that the prediction error would increase the further into the future that the prediction is looking.

Figure 15 is a histogram of the error values for the ID opcode training and test dataset LSTM predictions at time $t + 1$ when compared to the ground truth. This plot looks to be similar to a Normal distribution for both training and testing data.

An OOD dataset is created by generating 10,000 opcodes from a uniform distribution, i.e. non-baseline. Figure 16 is two histograms. The blue is the same ID error values plotted in Figure 15. The orange is the error histogram comparing the OOD predictions to the ground truth. It can be seen that the two distributions appear to be different.

In [19], the authors estimate parameters of a multi-modal Gaussian distribution to estimate the likelihood that a particular sample came from the baseline distribution. They do this by setting a likelihood threshold to make an ID/OOD decision. In this work, the use of SVM is explored to determine the boundaries between the ID and OOD distributions based on samples of P opcode

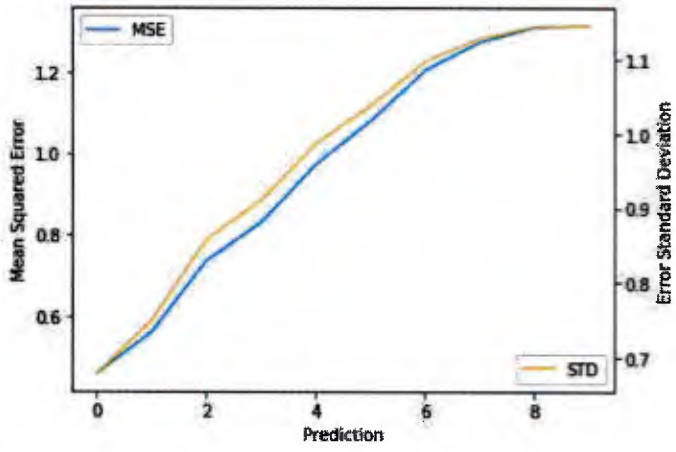


Figure 14: Mean squared error (blue) and mean error standard deviation (orange) over 3300 opcode inputs for predictions from $t + 1$ through $t + 10$.

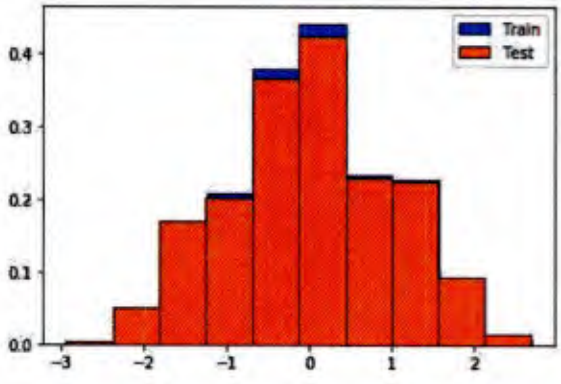


Figure 15: Histogram of the error values for training and testing data with sequence input, i.e. ID.

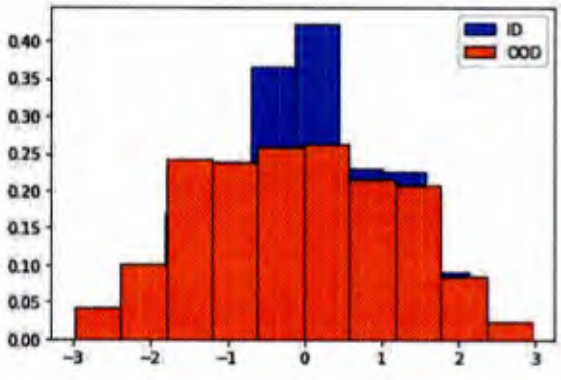


Figure 16: Histogram of the error values for testing data for ID, blue, and OOD, orange, data.

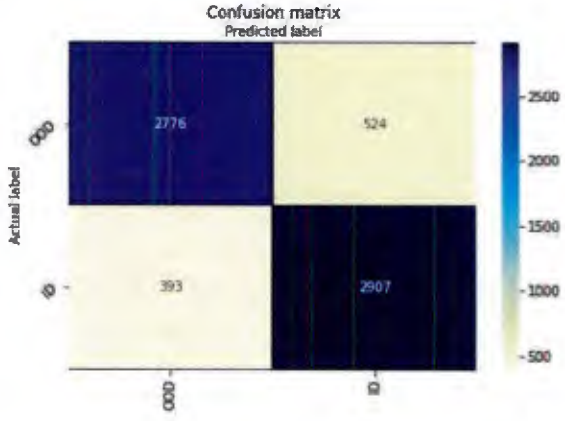


Figure 17: ID/OOD classification via SVM.

predictions. The training samples from both the ID and OOD datasets are labeled accordingly and an SVM is trained to make ID/OOD decisions. Figure 17 is a confusion matrix for the ID/OOD classification. The TPR is 0.876 and the FPR is 0.153. The results shown from these small-scale, tailored simulations appear to support further exploration of the LSTM-based baseline/anomaly detection process shown in Figure 12.

6 Conclusion

This report summarizes the work performed under the BioSec DaMIAN effort. The authors believe that the overall concept has been shown to have long-term potential which can be unlocked through additional research and the continued advancement of related technology. The relevant computational technology required to enable DaMIAN-like hardware trojan detection and mitigation will continue to be driven by commercial interests, which security-minded organizations will then be able take advantage of, as described in Section 4. Additionally, the use of RND for undocumented function detection and LSTM networks for IP core baseline development and anomaly detection both show promise but need to be scaled to address realistic scenarios. The authors would be interested in continuing this research in whatever way makes the most sense for DARPA and the DOD at large.

7 References

[1] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014.

[2] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 23–40.

[3] "The Immune System," https://www.hopkinsmedicine.org/health/conditions-and-diseases/the-immune-system.

[4] *How Does the Immune System Work?* Institute for Quality and Efficiency in Health Care (IQWiG), Apr. 2020.

[5] L. Hinzman, "Beating Video Games with Deep-Q-Networks," https://towardsdatascience.com/beating-video-games-with-deep-q-networks-7f73320b9592, Feb. 2019.

[6] W. C. Suski II, M. A. Temple, M. J. Mendenhall, and R. F. Mills, "Using Spectral Fingerprints to Improve Wireless Network Security," in *IEEE Global Telecommunications Conference (GLOBECOM)*. New Orleans, LA, USA: IEEE, Nov. 2008, pp. 1–5.

[7] G. Leopold, "FPGAs, ASICs Seen Driving Machine Learning," https://www.enterpriseai.news/2017/12/18/fpgas-asics-seen-driving-machine-learning/, Dec. 2017.

[8] "Xilinx AI Engines and Their Applications," Tech. Rep. Wp506, Jul. 2020.

[9] "Versal," https://www.xilinx.com/products/silicon-devices/acap/versal.html.

[10] T. Rondeau, "Domain-Specific System on Chip," https://www.darpa.mil/program/domain-specific-system-on-chip.

[11] J. Robertson, E. Wade, Y. Kopp, J. Bueno, and A. Hurtado, "Toward Neuromorphic Photonic Networks of Ultrafast Spiking Laser Neurons," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 26, no. 1, pp. 1–15, Jan. 2020.

[12] M. M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *ResearchGate*, vol. 27, no. 1, pp. 10–25, Mar. 2010.

[13] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, "A Survey on Machine Learning Against Hardware Trojan Attacks: Recent Advances and Challenges," *IEEE Access*, vol. 8, pp. 10 796–10 826, 2020.

[14] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using boolean functional analysis," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 697–708.

[15] H. Salmani and M. Tehranipoor, "Analyzing circuit vulnerability to hardware Trojan insertion at the behavioral level," in *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, Oct. 2013, pp. 190–195.

[16] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, Oct. 2013, pp. 471–474.

[17] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of Hardware Trojans and Maliciously Affected Circuits," *J Hardw Syst Secur*, vol. 1, no. 1, pp. 85–102, Mar. 2017.

[18] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by Random Network Distillation," *arXiv:1810.12894 [cs, stat]*, Oct. 2018.

[19] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," *Computational Intelligence*, p. 6, 2015.

[20] P. Hayton, S. Utete, D. King, S. King, P. Anuzis, and L. Tarassenko, "Static and dynamic novelty detection methods for jet engine health monitoring," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1851, pp. 493–514, Feb. 2007.

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[22] M. Basseville and I. Nikiforov, *Detection of Abrupt Change Theory and Application*. PTR Prentice-Hall, Apr. 1993.

[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[24] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, May 2009.

[25] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649.

[26] A. Sidorov, "Transitioning entirely to neural machine translation," Aug. 2017.

[27] B.-Y. Huang, H. Zhang, P. Subramanyan, Y. Vizel, A. Gupta, and S. Malik, "Instruction-Level Abstraction (ILA): A Uniform Specification for System-on-Chip (SoC) Verification," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 24, no. 1, pp. 1–24, Jan. 2019.

[28] P. Abad, P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J.-Á. Gregorio, "TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, May 2012, pp. 99–106.

[29] J. Lavina, "NIRGAM," http://nirgam.ecs.soton.ac.uk/.

[30] M. Amoretti, "Modeling and Simulation of Network-on-Chip Systems with DEVS and DEUS," https://www.hindawi.com/journals/tswj/2014/982569/, p. e982569, Apr. 2014.

[31] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, Mar. 1994.

[32] C. Olah, "Understanding LSTM Networks," https://colah.github.io/posts/2015-08-Understanding-LSTMs/, Aug. 2015.

[33] P. Sethuraman, "A Comparison of DNN, CNN and LSTM using TF/Keras," https://towards-datascience.com/a-comparison-of-dnn-cnn-and-lstm-using-tf-keras-2191f8c77bbe, Sep. 2020.

[34] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 2, pp. 869–904, Jan. 2020.

[35] "NVIDIA Turing: Reinventing Gaming Graphics," https://www.nvidia.com/en-us/geforce/turing/.

[36] "Autonomous Machines," https://developer.nvidia.com/embedded-computing, Oct. 2015.

[37] "Snapdragon 8 Series Mobile Platforms | Qualcomm," https://www.qualcomm.com/products/snapdragon-8-series-mobile-platforms.

[38] "Intel® FPGAs for Artificial Intelligence (AI)," https://www.intel.com/content/www/us/en/artificial-intelligence/programmable/overview.html.

[39] "FPGA vs. GPU for Deep Learning Applications," https://www.intel.com/content/www/us/en/artificial-intelligence/programmable/fpga-gpu.html.

[40] "Ascend 310," https://www.hisilicon.com/en/products/Ascend/Ascend%20310.

[41] "Federal Acquisition Regulation: Prohibition on Contracting With Entities Using Certain Telecommunications and Video Surveillance Services or Equipment," https://www.federalregister.gov/documents/2020/07/14/2020-15293/federal-acquisition-regulation-prohibition-on-contracting-with-entities-using-certain, Jul. 2020.

[42] "Kirin 970," https://www.hisilicon.com/en/products/Kirin/Kirin-flagship-chips/Kirin%20970.

[43] R. Daws, "Huawei debuts the Kirin 990 as the world's first SoC with AI and SA 5G," https://artificialintelligence-news.com/2019/09/06/huawei-kirin-990-world-first-soc-ai-sa-5g/, Sep. 2019.

[44] "DPU for Convolutional Neural Network," https://www.xilinx.com/products/intellectual-property/dpu.html.

[45] "1.4. Support Vector Machines — scikit-learn 0.22.2 documentation," https://scikit-learn.org/stable/modules/svm.html, 2020.

[46] "Sklearn.preprocessing.MinMaxScaler — scikit-learn 0.23.2 documentation," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[47] "Keras documentation: LSTM layer," https://keras.io/api/layers/recurrent_layers/lstm/.