Application of Machine Learning in Hardware Trojan Detection

Shamik Kundu Xingyu Meng Kanad Basu
Department of Electrical and Computer Engineering, University of Texas at Dallas
{shamik.kundu, xingyu.meng, kanad.basu}@utdallas.edu

2021 22nd International Symposium on Quality Electronic Design (ISQED) | 978-1-7281-7641-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISQED51717.2021.9424362

***Abstract*—Hardware Trojans (HTs), maliciously inserted in an integrated circuit during untrusted design or fabrication process pose critical threat to the system security. With the ever increasing capabilities of an adversary to subvert the system during run-time, it is imperative to detect the manifested Trojans in order to reinforce the trust in hardware. In this regard, Machine Learning (ML) algorithms, with their intrinsic capability to execute feature engineering at high learning rates, are emerging as promising candidates to be utilized by system defenders. In this paper, we explore Trojan detection mechanisms that are based on ML, and thereby investigate the prowess of the ML algorithms in bolstering system security. Furthermore, we analyze the efficiency of each proposed Trojan detection strategy based on the underlying ML algorithm. Finally, we underline some problems with existing Trojan detection approaches and discuss future research in the interest of improved performance of the employed ML algorithms, thus aiding in enhancing the intended hardware security.**

***Index Terms*—Trojan Detection, Machine Learning, Security.**

I. Introduction

The ever increasing proliferation of high-functioning hardware devices has led to a reduction in time-to-market of the large-scale Integrated Circuits (ICs). This has compelled the hardware vendors to incorporate third party intellectual properties and outsource the fabrication process [1]. As a result, unspecified identities get involved in the IC development cycle, thereby creating opportunities of Hardware Trojan (HT) intrusion in the circuit. A HT is a covert malicious alteration or inclusion that is aimed at modifying the intended function of an IC, or causing it to perform malicious functionalities. These Trojans are activated under a specific set of circumstances, as programmed by the attacker, thereby creating backdoor for sensitive information leakage, performance degradation or ensuing denial of service in critical applications [2]. For example, it was believed that, in 2007, a backdoor, built into a Syrian radar system was responsible for the system's failure [3]. With the fabless design trend in today's semiconductor industry, such covert Trojans pose serious security threats to highly sensitive industries including military and aviation, apart from consumer grade electronic devices. This imposes an urgent need in detecting the manifestation of Trojans and validating the trust in such commodity and sensitive hardware.

In this direction, several researchers have proposed Trojan detection mechanisms using thermal imaging, power monitoring and side channel analysis [2], [4]. However, with the increase in attacker's resources, adversaries are advancing the HT capabilities to trigger and unfold new attack dimensions. Therefore, detection techniques should be efficient in successfully identifying such unexpected attacks. Machine Learning (ML) algorithms, with their sophisticated mathematical modelling to process data with high complexity parameters, are being utilized to augment the performance of traditional Trojan detection algorithms [5]. The past decade has seen a meteoric rise of ML as the algorithm of choice for a host of popular applications belonging to the domains of computer vision, multimedia processing, graph, analytics, and search [6], [7]. This prowess of ML is being harnessed to bolster the domain of hardware security, where supervised and unsupervised models are trained on unique circuit signatures to detect potential attacks, as outlined in Figure 1. This paper explores such state-of-the-art defense techniques that apply ML for Trojan detection.

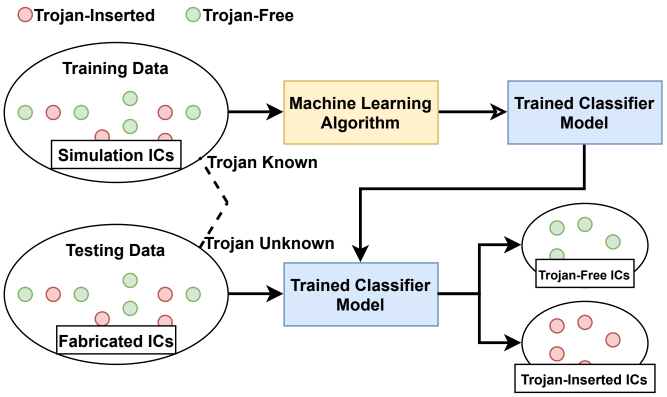

Fig. 1: Trojan Detection Flow Diagram.

The rest of the paper is organized as follows. Section II outlines the basics of HT, their detection strategies and traditional ML algorithms to be employed for the same. The popular ML-based detection techniques have been presented in Section III, and their performance have been demonstrated in Section IV. Finally, Section V concludes the paper.

II. Background

A. Hardware Trojans

In order to analyze the vulnerability of the system, existing research have developed various HTs. In general, a Trojan consists of two parts: trigger and payload. The **trigger** monitors various signals or a series of events in the circuit. The **payload** usually taps signals from the Trojan-free circuit and the output of the trigger. Once the trigger detects an expected event, the payload is activated to perform malicious behavior.

The payload remains inactive most of the time in order to avoid detection, and is only executed upon activating the trigger. A trigger can either remain always on, or can be triggered internally based on time and physical conditions, or externally, depending or an user input or a component output. When the trigger activates the payload, it can change the functionality of the device, degrade the performance, leak sensitive information or even result in denial of service [8]. The vulnerable locations of a Trojan manifestation in a circuit include processor, memory, input/output, power supply or clock grid. Such Trojans can be inserted by unspecified identities in the IC during specification, design, fabrication, testing or packaging phase of the development cycle [9].

B. Hardware Trojan Detection

In order to perform Trojan detection in an IC, two different scenarios may be prevalent — one that considers the availability of golden chips, and another that does not have access to the Trojan-free golden chips [10]. In either circumstances, detection is performed by classifying the features obtained from one of the following three sources, as shown in Figure 2.

*1) **On-chip Sensors Data**:* As the HTs are triggered in the circuit, their impact is manifested as inconsistencies in the low-level microarchitectural features such as hardware performance counters, data streams, electrical current measurements and power consumption traces. Features extracted from these on-chip sensors are then classified to efficiently identify the HTs in the circuit [11].

*2) **Netlist Data**:* In this method, a static analysis of the gate-level netlist is performed to extract a set of underlying features. These features majorly include a description of the logic fan-in cone, the number of gate levels from the output of a flip flop to the target net, the number of gate levels from the target net to the input of any flip flop and the number of gate-levels between the primary input to the target net and from the target net to the primary output of the design. These features are provided as input to a classifier to identify whether a particular net belongs to a Trojan-infected circuit [12].

*3) **On-chip Traffic Data**:* In an IC with multi-core communication system, Trojan-infected routers can divert the communication packets to destinations other than the intended destination, thereby leaking out sensitive information or launching denial-of-service attacks on the cores in the system. Trojans can also spoof a core in the circuit, which prevents other cores to access the packets delivered to the spoofed one. Hence, features extracted from this on-chip traffic can be efficiently classified to detect the activation of such Trojans [13].

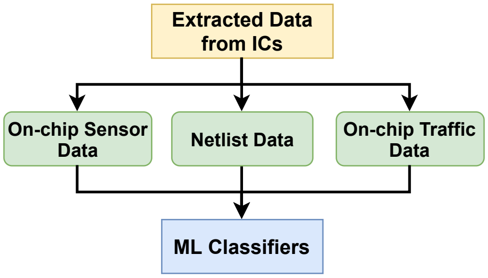

Fig. 2: Training Data Selection.

C. Machine Learning Algorithms

After the collection of meaningful data based on hardware behavioral analysis, efficient ML algorithms are trained on the dataset to detect the injection of Trojans in the circuit. Relevant feature selection will aid in improving the performance of Trojan detection, as well as implementing the detection framework on hardware with low overhead. ML can be majorly classified into two domains — supervised and unsupervised learning. Supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. Unsupervised learning, on the other hand, looks for previously undetected patterns in a data when pre-existing labels for the target classes are not available. Among a plethora of such ML models, the most widely adopted ones in the domain of HT detection are outlined in this section.

*1) **Support Vector Machine**:* Support Vector Machine (SVM) is a supervised learning model that efficiently classifies the incoming data with the help of a hyperplane [14], as represented in Figure 3a. The regularization parameter in SVM classifier scales down the error margin for improved performance on large data. There are different types of kernels used with SVMs, the most common of them being Radial Basis Function (RBF) kernel. The kernel dependent hyperparameters need to be fine tuned in order to extract optimum performance from the SVM classifier. One-class SVM, a category of SVM, uses the data from only one class to train the model [15].

*2) **Logistic Regression**:* Logistic Regression (LR) follows a supervised learning scheme, where the binary dependent variables are modeled along a logistic function. Since the function is sigmoid in nature, it follows an ‘S’ shape, as shown in Figure 3b. The LR classifier determines the optimal coefficient values from the training data through maximum-likelihood estimation approach [16].

*3) **K-Nearest Neighbors**:* K-Nearest Neighbor (KNN) is a supervised learning model that leverages identical data points that are closer to each other. Depending on the *nearest neighbor (K)* parameter value specified, K neighbors will be identified on the basis of distance metric from a specific query point. The KNN classifier furnishes the value with the most occurrence among the K labels as the output of the model [17]. Figure 3c outlines a KNN classification model.

*4) **K-means Clustering**:* K-means clustering is the most popular unsupervised machine learning algorithm that aims to partition n observations into K clusters, as demonstrated in Figure 3d. The K-means algorithm defines K cluster centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible [18].

*5) **Decision Tree and Random Forest**:* Decision Tree (DT) is a supervised learning model that has a tree structure with nodes and edges [19]. The DT classifier utilizes the nodes of the tree to test incoming values. Different learning algorithms with different branch splitting criteria, for example, CART [20], ID3 [19] or C4.5 [21] can be used for training the DT classifier. The Random Forest (RF) classifier consists of multiple individual Decision Trees that perform as an

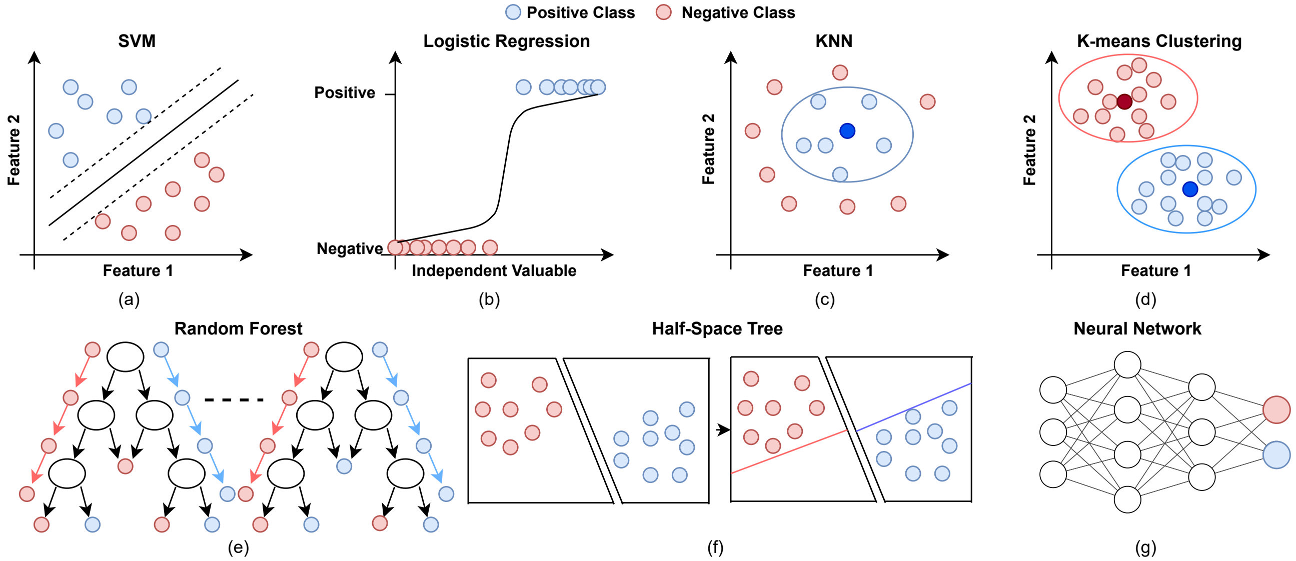

Fig. 3: Machine Learning Algorithms: (a) SVM (b) Logistic Regression (c) KNN (d) K-means Clustering (e) Random Forest (f) Half-Space Tree and (g) Neural Network.

ensemble, as shown in Figure 3e [22]. Each Decision Tree in the forest will furnish an output class, and the class with the highest number of predictions will furnish the output of the RF classifier.

*6) **Half-Space Trees**:* Half Space (HS) Trees are composed of an ensemble of binary trees that detect anomalies in data streams. As shown in Figure 3f, each HS-Tree partitions the dataspace into several windows and makes predictions based on the number of data points in each window [23]. For anomaly detection, most points should fall within the same window and those which do not are predicted to be anomalies.

*7) **Neural Networks**:* A Neural Network (NN) consists of an input layer, an output layer, and a number of hidden layers in between them. Neurons, having weighted input connections, a transfer function to combine inputs, and an output connection form the basic computational units in each layer [24]. During training, the back-propagation method updates the synaptic weights on the basis of a loss function, to obtain the optimum performance from the NN architecture. Figure 3g represents a neural network with two hidden layers.

III. Trojan Detection with Machine Learning

The capability of ML has been exploited to bolster the efficiency of traditional HT detection schemes. Existing research have proposed Trojan detection techniques for both golden chip-based and golden chip-free scenarios, that utilize the discussed ML algorithms.

A. Golden Chip-based Detection

The inconsistencies in parametric measurements as a result of an activated Trojan have been leveraged by ML classifiers for trust evaluation in a fabricated IC. A general framework for Trojan detection has been proposed in [25], where on-chip measurement acquisition sensors capture the electrical current traces in the circuit. The current is then converted to DC voltage to be classified by an one-class analog neural network to identify the untrusted circuit functionality. Trojans inserted during fabrication of an IC have been efficiently detected by analyzing the leakage current through a Bayesian inference-based technique, that is used to calibrate the process variation [11]. The power consumption traces in frequency domain are used to identify Trojans with a two-class SVM [26]. Similarly, relevant features have been extracted from power consumption traces and classified with a neural network to detect an activated Trojan in the circuit [27]. A run-time approach is proposed to detect HT in microprocessor cores by classifying performance counter data streams (cache and Translation Lookaside Buffer (TLB) miss rate) with HS-trees [28]. Along with performance counters, data from transmission power is also acquired to classify Trojans with an one-class SVM with RBF kernel [29].

HT in a circuit can also be identified by classifying all the nets in a netlist into Trojan and benign. In order to achieve this, [12] extracted 51 Trojan features from Trojan nets, and then identified the best set of 8 unique features to provide as input to the Random Forest classifier. The features include: (1) the logic fan-in gate up to 5-levels; (2) the number of flip-flops up to 5-levels away from the input and output of the target net; (3) the level of the nearest flip-flop to the input and output of the target net; (4) the number of multiplexers up to 5-levels away from the input and output of the target net; (5) the level of the nearest multiplexer to the input and output of the target net; (6) the number of m-level-loops where m is up to 5 from the input and output of the target net; (7) the number of nets that are assigned a constant value of 0 or 1 up to 5-levels away from the target net; and (8) the minimum number of gate-levels between the primary input to the target net and from the target net to the primary output in the design. Similar to [12], features such as logic gate input number, flip-flop input/output and primary input/output of gate-level netlist are extracted to perform an one-class SVM classification [30]. A multi-layer neural network is also applied on gate-level netlist to classify the Trojan-inserted design [31]. 11 features of the netlist are identified and used to train the model to detect the Trojan net manifested in the design.

Communication attacks triggered by HTs, such as core address spoofing, traffic diversion, route looping attack, can provide easy access to sensitive information or disrupt the functionality in many-core systems. In order to defend against such attacks, a run-time Trojan detection architecture for a custom many-core based on ML have been proposed in [13]. The dataset is generated based on the router behavior under normal and Trojan triggered settings. Features extracted from this on-chip traffic in both golden and Trojan-infected many-core chips are used to train SVM classifiers. Similar to [13], these features were utilized to detect HTs with the aid of supervised ML models, K-nearest neighbor (KNN), linear regression and Decision Tree [32], and unsupervised learning algorithms K-means clustering, Estimation Maximization, and hierarchical clustering [33]. In addition, these models can be updated to include the latest attacks with Modified Balanced Winnow (MBW) algorithm, which provide improved coverage against various types of Trojans [34].

B. Golden Chip-free Detection

Since the availability of golden chips is not guaranteed in all scenarios, existing research has demonstrated several attempts that do not require training data from Trojan-free circuits. In [35], multiple ML models such as Naive Bayes, Random forests, SVM, and logistic regression are applied to classify the transient power supply current data samples obtained from Monte Carlo simulations of ICs. The detection accuracy of each classifier is summarized and paired with the other classifiers to provide a higher coverage and accuracy on all types of Trojans. Statistical side channel fingerprinting is another popular HT detection method, wherein a parametric signature of a chip is collected and compared to a trusted region in a multi-dimensional space. This trusted region, comprising of side channel signal data, is developed through a combination of a trusted simulation model, measurements from Process Control Monitors (PCMs) which are typically present on a die, and advanced statistical tail modeling techniques. This signal data can be utilized to train an one-class SVM to classify a Trojan-inserted IC without having a fabricated chip [36].

A reference-free HT detection technique has been proposed in [37], that analyzes the controllability and observability in a gate-level netlist. An unsupervised k-means clustering analysis is utilized to demonstrate that the controllability and observability characteristics of Trojan gates furnish significant inter-cluster distance from those of genuine gates in a Trojan-inserted circuit, such that Trojan gates are easily distinguishable. An Optics density-based clustering learning is also proposed to detect Trojan in a netlist by classifying weakly correlated nodes or functionally isolated gates in a dendogram graph corresponding to the circuit under test [38].

IV. Performance Evaluation

In this section, the performance of each state-of-the-art Trojan detection technique is evaluated, as outlined in Table I. The evaluation metrics used to assess performances for each method introduced in Section III are discussed as follows:

a) ***Golden Design Required:*** This column represents whether a Trojan detection scheme is required to provide labelled training data for the machine learning model. Since the availability of golden design is not guaranteed, this might lead to lack of labelled data to the models. It enables the model to adopt a supervised or unsupervised learning approach. The unsupervised model might have a lower accuracy than supervised one, but can be used in more restricted situations.

b) ***Accuracy*** demonstrates the ratio of the testing data that has been correctly predicted by the model to the total testing data set, which serves as a major factor to evaluate the effectiveness and reliability of the ML model. The accuracy for a Trojan detection scheme can be represented as:

$$Accuracy = \frac{No.\ of\ (TP\ +\ TN)}{No.\ of\ (FP\ +\ FN\ +\ TP\ +\ TN)} \quad (1)$$

where, *TP* is True Positive, *TN* is True Negative, *FP* is False Positive, and *FN* is False Negative. For classifying an IC between Trojan (positive class) and Trojan-free (negative class), a True Positive is a Trojan correctly labeled Trojan while an application correctly labeled Trojan-free is a True Negative. A False Positive is a Trojan-free application incorrectly classified as Trojan, and a False Negative is a Trojan incorrectly classified Trojan-free.

c) ***True Positive Rate*** represents the efficiency of the model to correctly predict the Trojan-inserted design. A correct prediction of the HT will prevent the chip from performing unprecedented functionalities. A higher ratio of this means that the ML model is capable of detecting more number of Trojans-infected ICs, thereby bolstering the system security. The true positive rate can be represented as:

$$True\ Positive\ Rate = \frac{No.\ of\ TP}{No.\ of\ (TP\ +\ FN)} \quad (2)$$

d) ***False Negative Rate*** outlines the ratio of the incorrect classification of a Trojan into a Trojan-free design. A false negative prediction will ignore the Trojan-inserted design, causing unprecedented functionalities. This ratio should be ideally as low as possible to avoid misprediction of a malicious design. The false negative rate can be represented as:

$$False\ Negative\ Rate = \frac{No.\ of\ FN}{No.\ of\ (FN\ +\ TP)} \quad (3)$$

e) ***Training Features*** demonstrate the data features that are required by the ML model for training the network. These features, provided as input to the model are indicative of the stage of the development cycle in which the Trojan detection scheme can be applied. A scheme that can be applied in earlier stages will require less resource and time to detect the Trojan design, since it eliminates the cost to fabricate the design.

f) ***Training Data Size*** represents the sample size of the training data. The training data size will impact the accuracy

TABLE I: Performance of ML-based Hardware Trojan detection

Methods	Golden Design Required	Training Features	Training Data Size	True Positive	False Negative	Accuracy
[25]	Yes	2 Current Measurements	1k instances	>97%	0-2.8%	>98%
[28]	Yes	Architectural Events, Instruction Counts	1k samples	>90%	0%	>97%
[29]	Yes	6 Transmission Measurements	30	0-100%	0%	0-100%
[30]	Yes	5 Features of Netlist	Nets of 16 design	0-100%	0-22%	0-100%
[31]	Yes	11 Features of Netlist	Nets of 15 design	5.3-100%	0-2.8%	5.3-100%
[13]	Yes	8 Features of Runtime Traffic	716	N/A	N/A	85-99%
[33] unsupervised	No	8 Features of Runtime Data	216	N/A	N/A	50-87%
[33] supervised	Yes	8 Features of Runtime Data	216	N/A	N/A	85-99%
[32]	Yes	10 Features of Runtime Data	7260	N/A	N/A	67-98%
[35]	No	500 Current Samples	100	N/A	N/A	70-98%
[37]	No	Controllability, Observability Metrics	N/A	N/A	0%	N/A
[38]	Yes	2 features of Netlist	Nets of 8 designs	27-100%	N/A	>94%
[36]	No	6 Side Channel Fingerprints	Nets of 5 designs	100%	0-92.5%	>69-100%
[12]	Yes	11 features of netlist	Nets of 16 designs	5.3-100%	0-2.8%	>33-100%
[34]	Yes	2 features of Runtime Data	Nets of 17 designs	N/A	N/A	>80.2-91.1%
[27]	Yes	Power Consumption	27k Gates	N/A	N/A	N/A
[26]	Yes	Power Consumption	N/A	100%	N/A	N/A
[11]	Yes	Side Channel Signatures, Power Variants	Nets of 10 designs	>97%	N/A	>97%

TABLE II: Data used in various detection schemes.

Model	On-chip Data	Design Netlist	Runtime Traffic
SVM	[29]	[30]	[13], [26], [33]
Neural Network	[25]	[27]	[35]
KNN	—	—	[35]
Random Forest	—	[12], [31]	[35]
HS-Trees	[28]	—	—
K-means Clustering	—	[37], [38]	[36]
Logistic Regression	—	—	[35]

of the network. Therefore, a large unbiased dataset is likely to improve the accuracy of the ML model and vice versa.

The training data is fundamental to any ML-based approach. The model needs to be trained on the appropriate dataset and features to improve the prediction of the Trojan-inserted design. Although increasing the size of dataset could reduce the risk of over-fitting the model, different types of data will require certain pre-processing, thereby, increasing the cost and overhead. Post-development chip could produce very precise training data with on-chip sensors, which could provide a very accurate model such as [25], [28], [29]. However, it will lead to inevitable overhead and cost in chip development. Runtime data collection will address this challenge, but will suffer accuracy drop as demonstrated in [13], [26], [32], [34]. The netlist designs with very limited samples to train did not result in a low accuracy. On the contrary, it maintains a considerably high accuracy with enough features to balance the model and can be further improved with additional features such as [11], [12], [27], [30], [31], [35], [38]. Unsupervised training such as [33] shows a low accuracy compared to supervised training. However, this approach does not require golden chip to provide training data. This is a significant advantage since the availability of the golden chip is not always guaranteed, and it also indicates a more efficient data collecting process. [35]–[37] utilize different unsupervised ML approaches and provide a relatively high accuracy and low false negative rate without the golden design.

The features of the dataset demonstrate the attributes that are weighted in the ML model. Table II shows the data feature that prior works chose to train the detection network. The selection of domain-specific features will make a significant impact on the performance and accuracy of the model, as well as alleviate the difficulties and cost overhead to collect the data samples. [25] uses two current measurements collected from the on-chip sensors to generate data samples, similar to [28], [29]. This is the most effective way to create training features by dedicating specified parameters to classify the Trojan insertion. With a low overhead, this approach can be very effective as shown in Table I. Runtime power traces and data traffic are used in [13], [28], [29], [32], [33], which require no overhead to acquire the data. However, more features are needed to ensure correct classification. This might reduce the performance and increase the risk of over-fitting the model, in cases where some features are correlated to each other. Netlist features are also utilized in [30], [31] to classify the abnormal area of the netlist, thereby detecting the Trojan. This could help identify the Trojan designs at the early stage of the development, but limit the availability of dataset due to the lack of Trojan-inserted netlist. [27], [35], [37], [38] collect simulated features from the netlist design which further improves the size of training data. Unsupervised ML, as shown in [35], [37] take significantly more features to achieve similar performances as supervised learning. [36] used the features from runtime side-channel fingerprints to train the unsupervised model, incurring a massive feature space identical to [35], [37]. Training an unsupervised model is still a problem compared to supervised ones, albeit it can work in constrained environments when trained with appropriate features.

V. Conclusion and Future Direction

HT is an expanding research topic that has gained considerable attention over the last decade. Researchers have made significant progress in developing efficient algorithms for HT detection. In this article, we explored state-of-the-art techniques that employ ML for identifying the manifestation of Trojans in a circuit. Even though ML models furnish promising detection accuracy, it is rarely a near perfect 100%. This implies that some Trojans that are misclassified by the ML framework escape detection. This can be highly detrimental to the system security, as attacks launched by a single Trojan can disrupt the functionality of the circuit. Furthermore, recently adversaries have been targeting the ML models, both in software [39] and hardware domains [40]. Attackers can subdue classification process by injecting adversarial samples that are intentionally classified as benign while bearing malicious capabilities. Therefore, future research should not only improve the performance of the ML algorithms, but also focus on defending the models against potential adversarial attacks. Moreover, since most of the detection techniques encompass only Trojans inserted during design or fabrication phase, they should be bolstered by incorporating Trojans that can be inserted during specification design, EDA tool design, postmanufacturing test, and packaging process. This will aid in improving the overall reliability and security of the system.

References

[1] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. S. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection against hardware trojan attacks: Towards a comprehensive solution," *IEEE Design & Test*, vol. 30, no. 3, pp. 6–17, 2013.

[2] C. Rooney, A. Seeam, and X. Bellekens, "Creation and detection of hardware trojans using non-invasive off-the-shelf technologies," *Electronics*, vol. 7, no. 7, p. 124, 2018.

[3] S. Adee, "The hunt for the kill switch," *iEEE SpEctrum*, vol. 45, no. 5, pp. 34–39, 2008.

[4] L. Pyrgas, F. Pirpilidis, A. Panayiotarou, and P. Kitsos, "Thermal sensor based hardware trojan detection in fpgas," in *2017 Euromicro Conference on Digital System Design (DSD)*. IEEE, 2017, pp. 268–273.

[5] R. Elnaggar and K. Chakrabarty, "Machine learning for hardware security: Opportunities and risks," *Journal of Electronic Testing*, vol. 34, no. 2, pp. 183–201, 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[8] 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)*. IEEE, 2013, pp. 471–474.

[9] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, vol. 1, no. 1, pp. 85–102, 2017.

[10] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, pp. 1–23, 2016.

[11] X. Chen, L. Wang, Y. Wang, Y. Liu, and H. Yang, "A general framework for hardware trojan detection in digital circuits by statistical learning algorithms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1633–1646, 2016.

[12] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[13] A. Kulkarni, Y. Pino, and T. Mohsenin, "Svm-based real-time hardware trojan detection for many-core platform," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2016, pp. 362–367.

[14] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[15] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[16] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[17] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[18] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[19] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[20] W.-Y. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[21] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[22] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[23] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[24] M. H. Hassoun *et al.*, *Fundamentals of artificial neural networks*. MIT press, 1995.

[25] Y. Jin, D. Maliuk, and Y. Makris, "Post-deployment trust evaluation in wireless cryptographic ics," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 965–970.

[26] T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki, "Detection technique for hardware trojans using machine learning in frequency domain," in *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. IEEE, 2015, pp. 185–186.

[27] J. Li, L. Ni, J. Chen, and E. Zhou, "A novel hardware trojan detection based on bp neural network," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2016, pp. 2790–2794.

[28] R. Elnaggar, K. Chakrabarty, and M. B. Tahoori, "Run-time hardware trojan detection using performance counters," in *2017 IEEE International Test Conference (ITC)*. IEEE, 2017, pp. 1–10.

[29] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1506–1519, 2016.

[30] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware trojans classification for gate-level netlists based on machine learning," in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2016, pp. 203–206.

[31] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware trojans classification for gate-level netlists using multi-layer neural networks," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2017, pp. 227–232.

[32] A. Kulkarni, Y. Pino, and T. Mohsenin, "Adaptive real-time trojan detection framework through machine learning," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2016, pp. 120–123.

[33] A. Kulkarni, Y. Pino, M. French, and T. Mohsenin, "Real-time anomaly detection framework for many-core router through machine-learning techniques," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 1, pp. 1–22, 2016.

[34] V. R. Carvalho and W. W. Cohen, "Single-pass online learning: Performance, voting schemes and online feature selection," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 548–553.

[35] M. Xue, J. Wang, and A. Hu, "An enhanced classification-based golden chips-free hardware trojan detection technique," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. IEEE, 2016, pp. 1–6.

[36] Y. Liu, K. Huang, and Y. Makris, "Hardware trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proceedings of the 51st Annual Design Automation Conference*, 2014, pp. 1–6.

[37] H. Salmani, "Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 338–350, 2016.

[38] B. Cakır and S. Malik, "Hardware trojan detection for gate-level ics using signal correlation based clustering," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 471–476.

[39] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, pp. 43–58.

[40] S. M. P. Dinakarrao, S. Amberkar, S. Bhat, A. Dhavlle, H. Sayadi, A. Sasan, H. Homayoun, and S. Rafatirad, "Adversarial attack on microarchitectural events based malware detectors," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.