Netwise Detection of Hardware Trojans Using Scalable Convolution of Graph Embedding Clouds

Dmitry Utyamishev, *Student Member, IEEE*, and Inna Partin-Vaisband, *Senior Member, IEEE*

***Abstract*—Hardware Trojans (HTs) are malicious circuits that can be inserted into integrated circuits (ICs) during the design, manufacturing, or packaging phases. HTs can cause a variety of security and safety problems, such as data theft, denial-of-service attacks, and physical damage. A scalable reference-free framework is introduced in this article for netwise gate-level detection of existing and unknown HTs. The proposed framework consists of embedding-based automated netlist graph analysis and a supervised convolution-based classification of the individual IC net embeddings. A novel convolution algorithm for learning embedded IC graphs has been developed to overcome fundamental scalability limitation of existing graph convolutional neural networks. The performance of the proposed framework is experimentally demonstrated based on the TrustHub TRIT-TC benchmark suite, yielding a high recall of 96% and a precision of 86% for the *netwise* detection. The individual HT-compromised nets are highlighted within less than 0.1 s in >17,000-gate ICs. The proposed framework provides a scalable way to detect existing and unknown HTs without relying on HT-free reference ICs and can be effectively applied to large complex modern ICs. The unique combination of these characteristics makes the proposed framework more practical for real-world hardware cybersecurity applications as compared with prior art.**

***Index Terms*—Embedding convolution, gate-level netlists, graph embedding, hardware Trojan (HT) detection, machine learning (ML), supervised classification.**

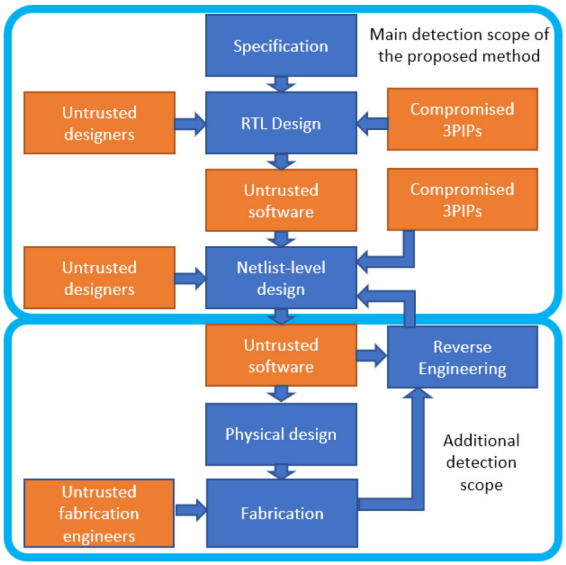

Fig. 1. Typical IC design flow. Unsolicited functionality can be seamlessly injected during almost any design or manufacturing stage. Potential vulnerability vectors are shown in orange. The proposed research threat model scope is shown with a blue frame.

I. Introduction

OUTSOURCING the design of integrated circuits (ICs) and manufacturing processes is a common practice to meet the challenging time-to-market requirements in the highly competitive semiconductor industry. Third-party intellectual property (3PIP) and predesigned in-house components allow semiconductor companies to reduce the IC design cycle cost-effectively. Outsourcing manufacturing allows small and large companies to avoid high-fab expenses and benefit from the latest fab technology. A downside of outsourcing design steps and relying on 3PIP is the risk of malicious and undocumented modifications of IC functionality. Unsolicited modifications can be unintentional (hard-to-verify circuit defects or anomalies) or malicious destructive hardware Trojans (HTs). While identification of those modified ICs is challenging, effective detection of HTs in practical ICs at the net-level resolution (i.e., netwise HT detection) is impractical with existing approaches.

Such modifications can be injected during any design stage, as illustrated in Fig. 1. At the register-transfer level (RTL) and netlist design stage, HTs can be injected as part of a compromised 3PIP block or by a malicious designer. During RTL or physical level synthesis, the system can be modified by untrusted third-party electronic design automation (3PEDA) software. Finally, an HT can be inserted into layout-level IC design files at a fabrication facility.

A typical HT comprises a malicious payload circuit, connected to existing IC nets, and is activated upon a triggering event. HT is designed to be triggered only upon a specific, statistically rare event to keep the malware undetected with standard functional testing (e.g., IC built-in self-tests). The complexity of a malicious payload varies according to the intended malware operations: from a single gate payload that can cause functional errors, degraded performance, IC shutdown, or denial of service to a complex payload circuit that can cause a high-level system malfunction by enabling software-based attacks (e.g., password theft) [1], [2]. However, a typical HT payload size is small enough compared with the

Manuscript received 13 September 2023; revised 6 February 2024; accepted 23 March 2024. Date of publication 29 March 2024; date of current version 20 September 2024. This work was supported in part by Google Research Scholar Award, and in part by the National Science Foundation under Grant CNS-2238976. This article was recommended by Associate Editor Y. Jin. *(Corresponding author: Inna Partin-Vaisband.)*

The authors are with the Department of Electrical and Computer Engineering, University of Illinois, Chicago, IL 60607 USA (e-mail: dutyam2@uic.edu; vaisband@uic.edu).

Digital Object Identifier 10.1109/TCAD.2024.3383348

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

overall IC. Thus, detection via physical inspection (including machine learning (ML) approaches [3]), human-expert manual analysis, or side-channel analysis is typically impractical due to IC design complexity, increasing process variations, and decreasing HT-to-IC size and noise ratio. While pre- and post-silicon HT detection methods have all been considered, early (i.e., presilicon) detection of HTs is expected to reduce security costs significantly and, thus, is highly desirable. We, therefore, leave the post-silicon HT detection out of the scope of this article, focusing on IC analysis at the gate-level design stages.

A new method for gate-level HT detection is proposed in this article. This method aims to determine the presence of malicious HTs in a gate-level netlist and identify at least one HT payload in those compromised ICs.

The method is effective with Trojans injected at the gate-level netlist or in earlier (e.g., RTL or high-level synthesis) design stages and compiled into the netlist level. The method can also be used to detect HTs injected later in the design process by reverse-engineering an IC into the netlist format and applying the proposed method. Thus, HTs inserted in the following attack scenarios can be detected with the proposed method: 1) HTs injected by a malicious rogue designer at any stage of the IC design process; 2) HTs injected by a 3PEDA software during a synthesis or physical design stage; and 3) HTs injected into layout-level IC design files (e.g., GDSII) before the fabrication stage.

II. Review of Existing HT Early Detection Methods

This research focuses on the gate-level detection of HTs. However, considering the modern synthesis software and recent state-of-the-art in IC reverse-engineering [4], the feasibility of transition among various levels of IC design abstraction makes certain HT detection approaches effective across several abstraction levels. For that reason, several broader (other than gate-level) state-of-the-art HT detection methods are included in the comparison.

Existing HT early detection methods are typically only effective for certain known HT types, require a golden HT-free IC for comparison, or are poorly applicable to large ICs. For example, automatic test pattern generation (ATPG) [5] has been proposed to increase coverage with functional analysis, and, consequently, the probability of HT activation. Formal verification methods have also been enhanced to enable the detection of high-level security flaws, such as information leakage [1] or unwanted register modification [2]. While the functional analysis does not guarantee detection, formal methods are typically limited to certain HT types. Both ATPG and formal verification methods exhibit poor scalability. Alternatively, recent code analysis methods [6] can flag the nets with low-activation probability by manually identifying security-sensitive RTL code lines. These methods, however, do not support sequential logic and often produce false-positive results even with full coverage. Several significant achievements in HT detection have been demonstrated with ML approaches. Based on an extensive literature review in the recent survey on ML against HT attacks [7], several ML-based HT detection methods focus on side-channel analysis because large volumes of side-channel data can be produced and utilized for training [8], [9]. However, these methods require a golden HT-free model to compare the reference side-channel data with the information obtained from the chip under investigation.

Another notable application of supervised ML for HT detection is a gate-level feature extraction and classification [10], [11] This approach is based on extracting a predefined set of features for each gate or a small subset of connected gates within an IC. The features are labeled and utilized for training a supervised ML model. While offering scalability and localization of HTs at the gate level, this method exhibits two primary limitations. First, relying on features of a single gate (or a small subset of gates), the full structural context of an integrated system is not considered. Second, malicious actors with prior knowledge of the extracted features can potentially "spoof" the HTs by manipulating the features to mimic patterns commonly associated with safe hardware (countermeasures for this type of attacks are actively researched in [12]).

Another promising HT detection approach is the graph-based static IC analysis. The guiding principle behind this method is exploiting the natural graph-like IC representation to learn certain system characteristics based on IC connectivity and statistical analysis, as demonstrated in [10]. In this work, eleven graph topology-based features have been manually determined and used to successfully detect HT nets, proving the feasibility of connectivity-driven HT detection. Additional examples of recent ML-based graph analysis are [13] and [14]. Piccolboni et al. [13] have trained a model to recognize HT patterns and utilized graph-matching algorithms to identify similar patterns in unseen ICs. However, exact graph matching is computationally hard, limiting the scalability of [13] with increasing IC and HT size. An approximate computation method has been proposed in [14] to address this concern. While HT detection in large ICs is possible with [14], this approach is still limited to known HT types and does not support the detection of unseen HTs. The limited availability of data for training ML models is yet another concern. While clustering algorithms have been considered for unsupervised HT learning, most existing ML-based HT detection approaches rely on supervised learning algorithms [15], [16], [17], requiring a golden reference IC [5] and a diverse high-quality training set. Such golden IC is often unobtainable, yielding along with robustness and scalability, another major limitation of existing ML approaches.

Finally, promising graph-convolution-based HT detection methods have recently been proposed [18], [19], [20] that support automatic extraction of graph topology information in an unsupervised manner, followed by supervised ML IC classification. These approaches allow to classify ICs into safe and HT-compromised but do not locate the HTs within a compromised IC. Thus, human expertise is still required to investigate each suspicious case. Poor scalability is yet another limitation of graph-convolution methods. Common implementations rely on memory-intensive graph convolution network (GCN) algorithms, exhibiting linear space complexity with the

graph size and number of GCN layers and quadratic space complexity with the number of features. Applicability of these methods is therefore limited to small 3PIPs. Recent advances in graph neural networks have enabled leveraging locality for efficient local subgraph sampling. By building GCN models based on the individual sampled subgraphs, space complexity is significantly reduced. This graph-convolution approach, coupled with modern graph sampling techniques, enables the efficient processing of 3PIP cores of any size and the precise localization of HTs with gate-level resolution. The overall design topology can be implicitly extracted with the GCN models, enabling HT detection without relying on predefined features. This capability makes GNN-based HT detection one of the most promising currently available techniques.

Albeit the continuous effort to mitigate HT risks, existing solutions are not robust, not scalable, and/or not reference-free. Furthermore, attackers have evaded every successful HT detection method with more advanced HTs. Thus, detecting HTs universally and effectively during the IC design process remains an urgent challenge. The following metrics are proposed for a fair comparison of HT detection methods and driving the development of an ultimate HT detection approach.

1) *Golden HT-Free Reference:* HT-free reference IC is typically required by traditional HT detection methods. Since HT may be injected in any phase of IC design, such a reference IC may not be available.
2) *Unknown HTs:* The ultimate detection method should be able to mitigate previously unseen HTs, since existing HT databases can be outdated or limited, and fundamentally new HTs continue to emerge.
3) *HTs With Sequential Triggering Mechanism:* Detection with existing approaches is often limited to combinational HTs. A sequential HT triggering mechanism poses different challenges as compared with a combinational trigger. A robust method is required to detect all types of active HTs.
4) *Human Engineered Features:* Methods that rely on predefined, manually selected sets of features are fundamentally limited to known HTs. These methods are expected to be mitigated by malicious designers, becoming obsolete with next-generation HTs.
5) *Resolution of HT Detection:* Most of the state-of-the-art methods only support an IC-level detection with IC classification into safe and HT-compromised. Such a low resolution of detection requires consequent expert intervention and extensive human engineer analysis at the sign-off to prevent false positives (FPs). A preferred detection method should exhibit a fine HT- or net-level resolution to minimize human intervention by localizing and/or explicitly identifying suspicious parts (e.g., payload nets and/or HT triggers) of the integrated system.
6) *Scalability With IC Size:* Most existing approaches exhibit limited scalability. The time and/or space complexity of underlying algorithms does not allow the practical use of state-of-the-art methods with larger integrated systems and/or lower-IC design abstraction levels.
7) *Overall Design Structure Extraction:* Current HT detection approaches primarily focus on those features extracted from the individual gates or a small subset of gates, disregarding the topological system-level information - a potential cornerstone in identifying malicious system components. Alternatively, some advanced HT solutions utilize mechanisms for iteratively exchanging messages between connected nodes. This approach allows nodes to propagate the information through the graph structure, effectively extracting and integrating features from all the surrounding context. However, these features cannot be exported from the model and used for further analysis. The ability to explicitly export features that capture the local and overall circuit topology facilitates fundamentally novel research directions in HT detection, including visualization and classification with a broad range of methods.

An effective HT detection framework should not only show great quantitative detection metrics but also consider and address all the aforementioned qualitative metrics. Modern top-performing HT detection solutions are listed in Table I with respect to the seven metrics, comprising a broad spectrum of detection principles. Note that each method in the table exhibits at least one limitation. Thus, despite the reported high-quantitative detection performance, modern state-of-the-art solutions lack the critical traits to become ultimate, reliable, and scalable commercial solutions.

III. Proposed Framework

In this article, a framework for HT detection is proposed. The framework leverages unsupervised extraction and supervised classification of information based on an IC netlist graph. The framework consists of 1) an analytical model that encodes the triggering frequency of individual nets as Euclidean distances; 2) an HT-aware knowledge graph embedding (KGE) algorithm; and 3) multiple detection policies for netwise classification based on embedded IC vectors.

The overall HT detection process is divided into two distinct phases. During the first phase, an IC is transformed into a graph and consequently into an embedding cloud representation. In the second phase, connectivity information from both the graph and embedding cloud is processed using policies based on different classification principles. Finally, a resulting classification output (safe or compromised) is aggregated from multiple intermediate policies. It should be noted that policies can be executed simultaneously (if they rely solely on IC graph and embedding data) or organized as a directed acyclic graph pipeline with producer-consumer relations. In the latter case, upstream policies generate auxiliary information that is utilized by downstream policies. This approach is illustrated in Fig. 2.

Several HT detection policies, including search space reduction, feature engineering, and convolutional neural network (CNN), are investigated for classifying the individual embedded nets in the latent space and compared. Policies that address all the metrics, as listed in Table I, are proposed. The concept of KGE for HT detection was first introduced

TABLE I
QUALITATIVE EVALUATION OF CURRENT STATE-OF-THE-ART REFERENCE-FREE HT DETECTION METHODS

	Detection level	Unknown HTs	Sequential HT detection	Feature engineering	Resolution	Scalable with IC size	Structure extraction
ISCAS'23 [21]	Gate	Yes	Yes	Automated extraction	Net	Yes	Implicit
HOST'21 [20]	Gate	Yes	Yes	Automated extraction	IC	No	Implicit
TCAD'21 [18]	RTL, Gate	Yes	Yes	Automated extraction	IC	No	Implicit
AsianHOST'18 [16]	RTL	Yes	Yes	Predefined, human-engineered	IC	Yes	No
ICCIT'19 [22]	RTL	Yes	No	Predefined, human-engineered	IC	No	No
TECS'17 [13]	RTL	No	Yes	Predefined, human-engineered	Net	No	No
ISCAS'19 [15]	RTL	No	Yes	Predefined, human-engineered	Net	Yes	No
ISCAS'17 [10]	Gate	Yes	Yes	Predefined, human-engineered	Net	Yes	No
IOLTS'21 [11]	Gate	Yes	Yes	Predefined, human-engineered	Net	Yes	No
This work	**Gate**	**Yes**	**Yes**	**Automated extraction**	**Net**	**Yes**	**Explicit**

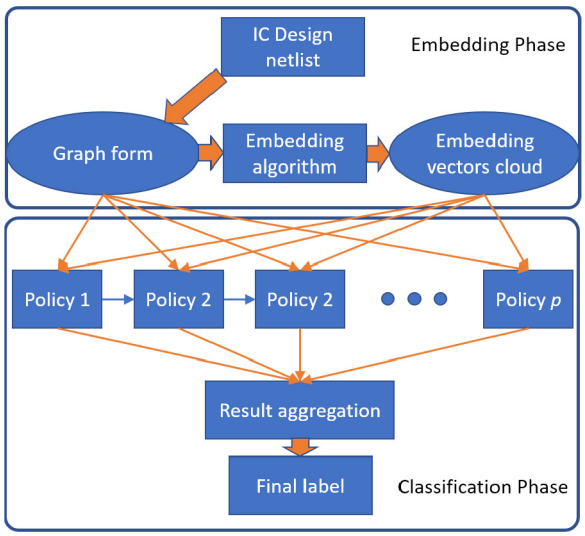

Fig. 2. Proposed two-stage detection methodology. IC graph and corresponding embedding space are generated in the first stage and processed with multiple detection policies in the second stage.

in our recent work [23]. However, the model in [23] only supports a reduction in the search space for existing third-party HT detection tools. The high-false-positive prediction rate prevents the model in [23] from being used as a stand-alone HT detection tool. The approach proposed in this article is based on two-stage semi-supervised classification. First, a graph constructed from a netlist-level IC is embedded into a cloud of vectors with unsupervised KGE. Consequently, the embedded nets are classified as safe or malicious based on the convolved information of the coordinates of the individual and semantically related net embeddings. The proposed HT detection pipeline and major components, including unsupervised graph embedding and supervised classification, are shown in Fig. 3.

The detection flow comprises unsupervised graph embedding and supervised netwise classification. An input netlist-level IC representation (1) is parsed into a graph representation and stored (2) with its corresponding metadata information. The graph information is read (3) by the graph embedding algorithm and processed into an embedding vector cloud (4), which is labeled with unique identifiers to preserve the relationship with the corresponding graph nodes. The embeddings are then filtered (5), and vectors, which are considered potentially suspicious, are tagged for future usage. Tagged embeddings are processed with a human-engineered detection policy (6), and the resulting label is saved for future use. Finally, tagged data is sampled (7) into a perception field feature array using the embeddings and graph information. The perception field of each embedding is then classified (8) with a supervised model, and the per-embedding predictions are aggregated (9) according to the preferred detection policy. Using a zero-tolerance detection policy, the overall IC is labeled as compromised if an HT embedding is detected. The resulting classification labels are reported (10) for the entire IC and the individual suspicious nodes. Further details on the proposed unsupervised embedding of an IC netlist and supervised classification of the embedded cloud are provided in the following subsections.

1) Unsupervised Graph Embedding: Consider the representation of a netlist-level IC as a graph in which the IC nets and input-to-output IC gate relations are mapped onto, respectively, the IC graph nodes and edges. The IC connectivity information, as defined by the graph, is embedded into a latent vector space in which the Euclidean distance between two embedded nodes represents the conditional probability of triggering one of the nodes by the other node. Thus, two net embeddings are adjacent if activation of one of the nets frequently triggers the other and are farther away if the conditional triggering is rare. The statistics of triggering relations between IC nets is analytically described in this article in the quantitative form of geometric distance and implemented within the ComplEx [24] knowledge graph algorithm, as explained below.

The relation vector g between two graph nodes k_i and k_j is encoded as a value $A[k_i, g, k_j]$ of a rank 3 tensor. The tensor value is approximated with a dot-product function, $f(\cdot)$, of the embedded vectors and the corresponding relation vector, g. First, IC nets (i.e., graph nodes) and nets' relations

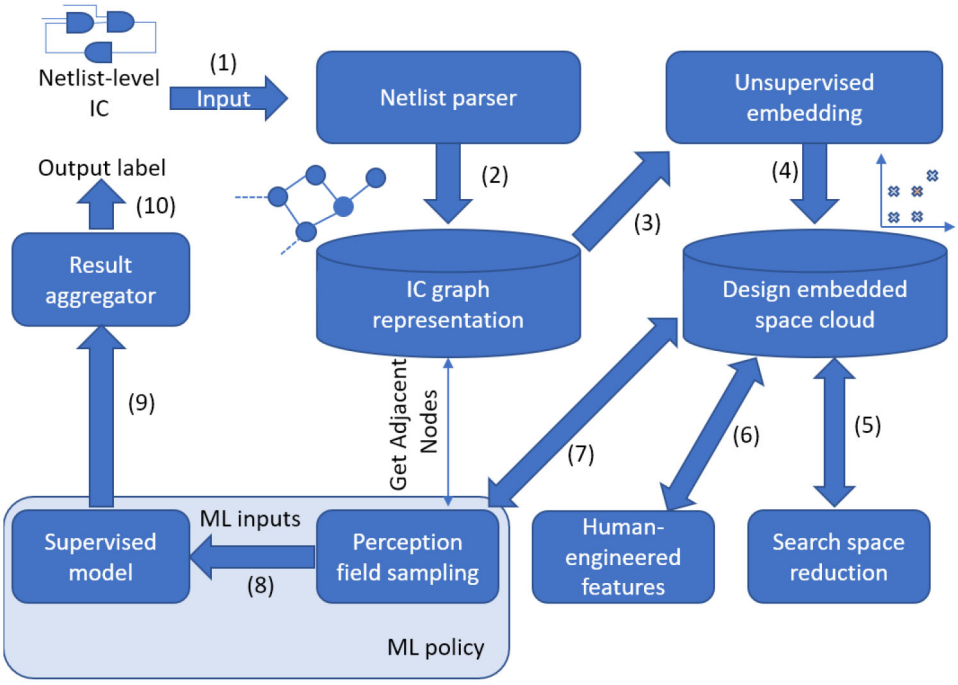

Fig. 3. Proposed supervised HT detection pipeline. An IC is converted into a graph representation and embedded as a cloud of vectors in a latent space. The individual embedded nets are evaluated for association with an HT.

(i.e., graph edges) are randomly embedded as, $u = e(\cdot)$. Next, the stochastic gradient descent (SGD) algorithm is executed to determine the embeddings that minimize the loss between the $f(\cdot)$ and those known values of the tensor A (e.g., the values of relations/edges between the adjacent nodes are initially known). As a result, the value of $f(u_{k_i}, u_g, u_{k_j})$ increases/decreases with stronger/weaker relation g between entities k_i, k_j. Since embeddings are optimized to maximize their dot-product $f(\cdot)$, the embeddings of nodes with high connectivity have similar values, and therefore are placed close to each other in the embedding (latent) space. Alternatively, embedded vectors of nodes with no relation are placed maximally apart, minimizing the cross-product. This approach automatically detects topological artifacts and extracts data without explicit feature engineering and without relying on labeled data. Note that ComplEx time and space complexity is linear with input size. The model can be enhanced with graph partitioning, multithreading computation, distributed execution on disjoint graph parts, and batched negative sampling (see BigGraph [25] for example) to further increase embedding scalability with IC size, up to the constant space complexity.

The correctness of the proposed embedding has been visually evaluated in our recent work [23], exhibiting a cluster of rarely triggered nets embedded in the center of the latent space. While significantly reducing the HT detection search space, the method in [23] exhibits low-precision values, and, therefore, cannot be used to identify HTs without auxiliary HT detection tools. In this work, a KGE-complementary HT detection method is developed based on spatial characteristics of the embedded IC vectors. For example, vectors in embedded space can be processed with geometrical or statistical methods. Three KGE-complementary policies are proposed and investigated in this work, as described in the following subsections.

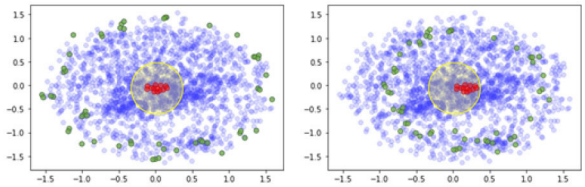

Fig. 4. Illustration of embedding of an HT-compromised IC in a latent space, explored in [23]. Embeddings associated with the HT payload, triggering nodes, and all the other nodes are highlighted in, respectively, red, green, and blue. The geometric center of the embeddings' cloud is shown in yellow covering a fraction of all the embeddings. The HT is triggered by (a) rarest and (b) more common combination of system events.

A. Search Space Reduction

This policy aims to mark HT-suspicious embeddings and their corresponding IC nets based on computationally easy high-recall metrics, reducing the search space for the downstream detection policies. Based on the preliminary results [23] illustrated in Fig. 4, the embeddings of nets with low-triggering probabilities are distributed farther away from each other, along the perimeter of the latent space. Alternatively, HT payload nodes exhibit zero-order proximity with rarely triggered nodes. Thus, HT payload embeddings are expected to be located in the geometric center of the embedded latent space, minimizing the Euclidean distance to all the low-triggering probability embeddings.

In practice, an IC can comprise multiple weakly related functions (e.g., different high-cohesion, low-coupled functional blocks), resulting in a multimodal distribution of embedding vectors. Thus, a single high-risk cluster in the center of the embedded space, as assumed in [23], cannot effectively characterize a general IC with a multimodal

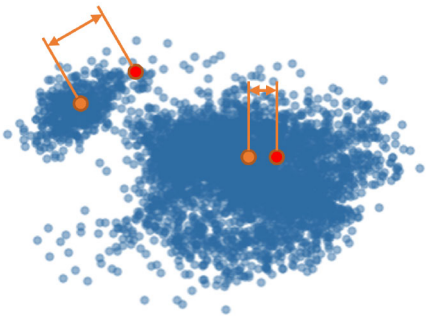

Fig. 5. Illustration of the distance metric between the embeddings under consideration (red) and the closest cluster centroid (orange). A typical case of a multimodal embedding distribution is shown, with the major (largest) cluster on the right, and a smaller secondary cluster on the left.

embedded vector distribution. Another challenge is related to the granularity of the clustering algorithm, which often requires manual hyperparameter tuning to produce clusters of desired size in various IC embedding clouds.

Intuitively, HT payload embeddings are placed in the centers of corresponding subclouds embedded from the individual functional blocks, rather than the center of all the embeddings in the latent space. To evaluate this assumption, the distribution of distances to the closest centroids is evaluated using 370 HT ICs from the TRIT-TC benchmark suite. The distances from HT embeddings to the closest cluster are shown as a histogram in Fig. 6, where orange histogram bars represent the monomodal distribution case, and blue histogram bars represent the multimodal distribution case. The reference HT dataset analysis shows that the distances from HT embeddings to the closest cluster centroids are less than 0.13 of the embedding cloud radius, and therefore, suspicious nets can be thresholded with this radius-based metric. To exploit this HT characteristic, splitting IC embeddings into the major (largest) cluster and secondary (smaller) clusters of functional blocks is proposed. First, all the embeddings, $U = u$, are grouped using the OPTICS [26] clustering algorithm based on their spatial distribution in the latent space. Consequently, suspicious net embeddings are marked based on their distance to the closest cluster centroid in the following manner. The distance to the closest centroid, $d(u)$, is calculated for each net embedding, $u \in U$

$$d(u) = \min_{u_c \in C} \left(\frac{|u - u_c|}{d_U} \right) \tag{1}$$

where $d_U = \max_{u_i, u_j \in U}(|u_i - u_j|)$ is the diameter of the embedding cloud and C is the set of the centroids of all the clusters, as determined with the OPTICS algorithm. If the value of $d(u)$ is less than a certain radius threshold, the embedding u is marked suspicious. A high-radius threshold value is preferred to eliminate more non-HT embeddings at the expense of potentially missing some HT embeddings.

1) Human-Engineered Spatial Pattern Detection in the Embedding Cloud: With the proposed IC KGE, geometrical information in the latent space can be efficiently exploited to extract high-level IC insights [23], determine unique IC characteristics, and detect IC anomalies. For example, embeddings of nets that trigger rare events (e.g., HT triggers) are often embedded farther away from all other embeddings in the latent space and are, therefore, located sparsely and uniformly at the periphery of the cloud, as shown in Fig. 4. Alternatively, rarely triggered nets (e.g., HT payloads) are typically embedded at the center of the cloud, at the minimum distance from their triggers embedded along the periphery. Since an HT trigger and payload in an IC are always adjacent directly connected nodes (i.e., first-order proximity neighbors), HTs can be distinguished and classified based on the mutual location of those uniquely placed embeddings (a.k.a. spatial pattern). The proposed KGE algorithm forces adjacent node embeddings to be placed close to each other, forming dense local clusters in the latent space. This is because two adjacent nodes have a shared edge and often share neighbors. This spatial pattern has been visually confirmed in some combinational ICs from the TRIT-TC benchmark suite. Since safe nets are often observed to be embedded as dense, uniformly distributed local clusters, nodes placed at a larger distance from their neighbors are considered to be outliers and HT-suspicious. One reason an outlying node is placed farther from its neighbors is the node connection with a rarely triggered event. A common configuration of a set of safe neighboring embeddings with and without an outlying HT-suspicious embedding is shown in Fig. 7. The distance to the outlier embedding node from the cluster centroid, as compared with the cluster radius, is an important characteristic of the HT detection process.

These properties can be expressed in the form of the following score metric with respect to a node k_0 embedded as a multidimensional vector u_0:

$$\text{score} = \frac{\max_{j \in [1, n-1]} |u_j - C_n|}{(n-1)|u_{k_0} - C_n|} \tag{2}$$

where C_n is a dense cluster centroid

$$C_n = \frac{\sum_{i=0}^{n-1} u_i}{n-1} \tag{3}$$

and u_i is an embedded node in the form of a multidimensional vector, n is the number of the k_0's first-order proximity neighbor nodes, u_{k_0} is the ordered list of neighbor's embeddings sorted by their L2-norm with respect to k_0.

The information about the distribution of the proposed score can be gathered in safe in the available benchmark data for safe and HT embeddings. The threshold value, distinguishing scores of different embedding classes, can be estimated empirically based on existing labeled data.

This heuristic is an example of manual feature extraction with a predefined detection policy. As expected, it works well for certain HTs but fails to detect HTs of other types. Combining various features and policies can increase the effectiveness of a heuristic, but practically, formalizing HT properties into a form of mathematical expression of a robust computer algorithm is highly challenging for human engineers. Alternatively, HT designers can consider these metrics to design new types of HTs, undetectable with existing HT countermeasures. To alleviate these issues, supervised ML methods should be considered.

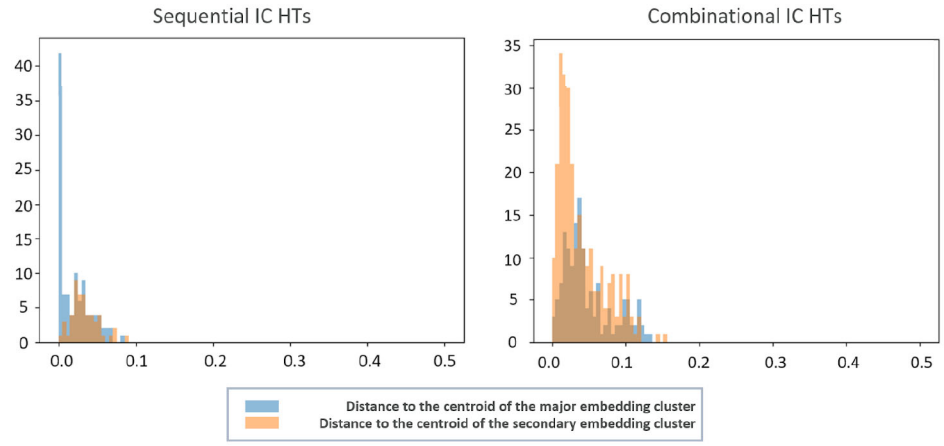

Fig. 6. Distance between the HT embeddings and cluster centroids as determined based on 370 analyzed TRIT-TC ICs. Blue histograms illustrate the distance in the case of unimodal distribution, and orange histograms illustrate the multimodal distribution case. In all cases, the distance to the closest centroid is less than 0.13 of the centroid radius, exhibiting a concentration of HTs near centroids of embedding clusters.

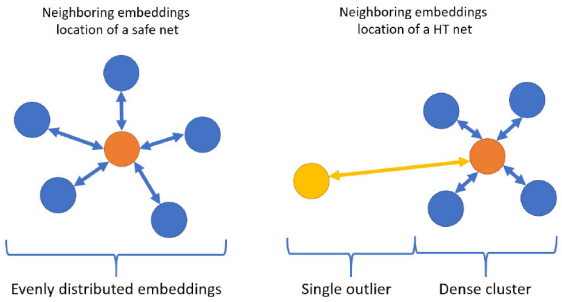

Fig. 7. Embedded node (orange-shaded centroid) and a typical distribution of its embedded neighboring nodes, as visualized in a 2-D latent space. Embeddings of the neighboring nodes are shown in blue, forming topologically different configurations with uniformly distributed embeddings around the centroid and (a) no outliers in a safe case and (b) single yellow-shaded outlier that corresponds to the embedding of an HT trigger in the compromised case.

B. Supervised Embedding Convolution

To mitigate the high-memory complexity with existing GCNs, a robust supervised policy is proposed for detecting proximity-based embedding spatial patterns. A convolution network is designed to classify graph nodes (and consequently, the individual IC nets) as safe or malicious, based on their topological relationship with other graph nodes and the co-location of the corresponding embeddings.

The 2-D grid convolution is commonly utilized for image processing applications due to its ability to extract local characteristics. An important parameter in CNNs is the receptive field. The receptive field is a small fragment sample of an input image which is used to traverse a 2-D grid in a left-to-right top-to-bottom direction. The receptive field contains a fixed number of values, e.g., as captured by a 3×3 matrix (see Fig. 8). In the context of image processing, each value corresponds to a specific pixel's color value.

CNN image processing is based on evaluating the similarity of an image region with a known pattern (a.k.a. convolutional filter). This is accomplished by computing the dot product of the values within a receptive field and CNN filtering kernel (filter). The resulting numerical value indicates the presence or absence of the corresponding pattern in a given image region. To practically calculate the dot product, the values of a multidimensional receptive field and filter matrices are rearranged in an isomorphic manner. This is a simple task with 2-D images, for which the matrices' values are rearranged row by row (i.e., flattened in a row-major order) into corresponding 1-D vectors, as exemplified in Fig. 8.

Intuitively, the problem of pattern-based HT detection in a latent vector space resembles typical image processing, and thus, CNN-based policies are a promising solution. However, unlike image data, the neighbors of a node in a graph are unordered and variable in size. To compare two arbitrary subgraphs using convolution, an isomorphic transformation of a subgraph is required, i.e., a perception field in the form of a subgraph needs to be sampled, and its nodes need to be *systematically* rearranged, yielding similar outputs with similar subgraphs. However, an isomorphic transformation of graphs is a primary challenge in graph theory since an arbitrary subgraph typically comprises a random number of neighbor nodes, lacking a well-defined order.

A canonicalization process is required to obtain an isomorphic graph (i.e., a different form of graph with the same number of vertices, edges, and edge connectivity) with a fixed ordering of its nodes. While every two isomorphic graphs have the same canonical form and can be converted into each other with a canonicalization process, this is an NP-hard problem and, thus, computationally infeasible for practically sized ICs. Therefore, processing an arbitrary graph with a convolutional network is a complex problem and the primary limitation for efficient utilization of spatial-based graph convolution for IC graph processing. To mitigate the complexity of isomorphic transformation in graphs, *approximate* graph canonization is proposed in this work. This approach exploits spatial information of embedded IC graphs for efficiently rearranging the receptive field, enabling the convolution of graph node neighborhoods of arbitrary sizes. To the best of the authors' knowledge, this is the first approach that exploits the spatial properties of the embedded space for the netwise-classification of IC behavior.

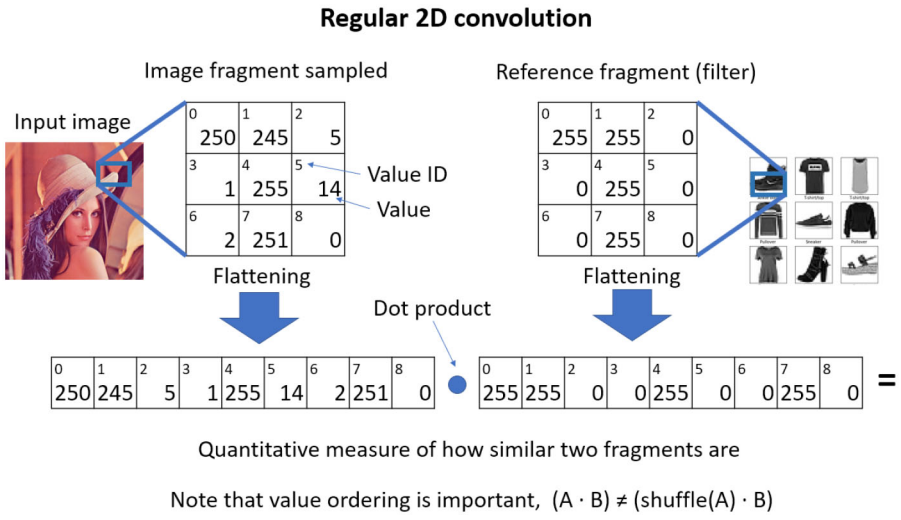

Fig. 8. Illustration of a 2-D convolution. The CNN receptive field is formed by sampling pixel values in a small rectangular area. Flattening and multiplying the values of the receptive field and CNN kernel produces information about the presence of a feature in a given image fragment.

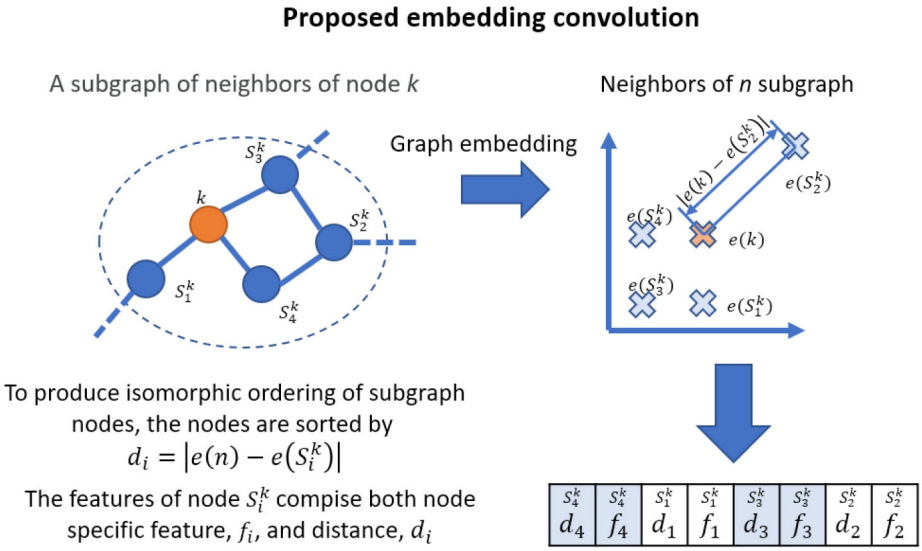

Fig. 9. Proposed process of designing a stable receptive field for embedding-based graph convolution.

The proposed approximate graph canonization is based on the following property: the distance between node embeddings (which represents the semantic proximity between the nodes in the original graph) is fixed across various KGE runs for a given IC, i.e., distances between various net embeddings tend to saturate to similar values. As a result, neighboring nodes, sorted by the L2 distances to a common node, represent a stable and unique mapping of neighboring subgraphs, thus exhibiting graph isomorphism, as illustrated in Fig. 9.

To generate a receptive field of size N around a graph node k, the graph is traversed in a breadth-first search (BFS) until N nodes are discovered. Next, the BFS algorithm is continued until the ongoing traversing deep level is complete. Finally, the subgraph of all the discovered nodes S_k of size $N_d > N$ is flattened into an ordered list, sorted by the $L2$ distance to the node k, and truncated to size N. Each value of the ordered list is used as the CNN receptive field value, as shown in Fig. 9. Local neighborhoods of the target nodes are, therefore, represented as feature vectors. These feature vectors are similar to the original nodes in terms of the spatial features embedded within the neighboring nodes and in terms of local neighborhood topology. These similarities can be captured through supervised training and utilized as features to differentiate HT hardware components.

IV. Experimental Results

In this section, the experimental setup is explained and the results of the evaluation of the proposed framework are provided. Prototyping details are provided for benchmark preprocessing, software frameworks, and ML algorithms, including the parameters for graph embedding, embedded space dimensionality reduction, and the proposed supervised embedding convolution HT detection. Results are explained and compared with the state-of-the-art.

A. Dataset

TrustHub TRIT-TC benchmark suite, as generated by a gate-level Trojan insertion tool (TRIT), is utilized to demonstrate the proposed HT detection framework. A total of 260 combinational HTs injected into four different ICs and 110 sequential HTs injected into four other ICs are used for performance evaluation. The netlist-level benchmark ICs are parsed into the text graph representation format, producing a list of three-value tuples, denoted as LHS-node, relation-type, RHS-node. Each tuple describes the connection between two nets (LHS-node and RHS-node), as facilitated by a gate of a specific type (relation-type). This format is native to the proposed graph embedding framework (see BigGraph [25] and KBC [27] documentation). The meta-information available with the IC components, such as net names and HT labels, is used for evaluation but not for classification.

B. Netlist Embeddings and Convolution Parameters

State-of-the-art ComplEx KGE algorithm [24] is utilized to embed a benchmark IC graph as defined by the parsed list of nets and net relations. The algorithm is executed with the following hyperparameters:

1) `factorization_rank = 100`;
2) `regularization_method = N3`;
3) `regularization_weight` $= 10^{-5}$;
4) `batch_size = 256`;
5) `optimizer = "Adagrad"`;
6) `learning_rate = 0.1`; and
7) `number_epochs = 100`.

Note that with ComplEx algorithm, data is embedded into complex-valued vectors. Thus, the effective dimensionality of the embedded space is twice the factorization rank. In this article, the resulting 200-D embedded space is further projected into a 2-D space using the T-SNE algorithm [28] for debugging and visual analysis. The following parameters are used for T-SNE projection:

1) `n_iterations = 1,000`;
2) `perplexity = 100`; and
3) `learning_rate = auto`.

Features extracted from both the 200- and 2-D spaces are used for convolution.

To filter the suspicious nets, OPTICS clustering algorithm is used with the following parameters:

1) `min_sample = 20`;
2) `cluster_method = "xi"`; and
3) `xi = 0.05`.

If the distance from the centroid to a node under test is greater than 0.13, the node is labeled as safe and excluded from the consequent HT detection process.

The following parameters are used to build the embedding convolution receptive field: the number of nodes per receptive field = 10, including distance per each node = True. The receptive field size is determined with a cross-validated parameter optimization search.

For the supervised classifier, the following layers are used: Linear (input size = 20, output size = 40), ReLU, Linear (input size = 40, output size = 80), ReLU, Linear (input size = 80, output size = 2), Sigmoid. Binary cross entropy is used as a loss function, and SGD (learning rate = 0.1) is used as an optimizer.

C. Data Generation for Supervised Learning

Data for the proposed supervised model training is generated using the following method. An IC from the TRIT-TC benchmark suite is parsed into its graph representation. The IC graph is transformed into a cloud of embedding vectors using the proposed graph embedding algorithm, as described in Section III. For each embedded vector, a set of features is automatically extracted based on connectivity information (as described in Section III-B). This process is repeated for all the individual nets in each IC benchmark. The true labels for the individual embedded vectors are automatically determined based on the meta-information from the original netlist file—if a net name includes the substring "troj," the HT label is set to 1 (corresponding to an HT-net). Otherwise, it is set to 0.

A total of 80% of the generated data is used for training, and the remaining 20% is used as a test set in a k-fold manner. This setup enables the demonstration of unseen HT detection without an HT-free reference IC. Ten different training sets are generated and combined to ensure that the detection does not depend on a random seed. The unsupervised embedding algorithm is executed with a different random seed for each of the ten training sets.

D. Performance Metrics

To evaluate the performance of the proposed method, an HT-free (safe) version of IC is generated for each HT-injected (compromised) IC by removing the HT nets from the netlist file. Features for the supervised model are extracted for suspicious nets using the previously described flow: the IC under test is parsed into a graph form, the netlist graph is embedded, the convolution receptive fields are sampled for each suspicious net, and the suspicious nets are classified with the supervised model. If a net is classified as an HT net (supervised model output is greater than 0.5), the IC is labeled as compromised. If none of the nets is classified as an HT net, the IC is labeled as safe.

The IC-level prediction is considered true positive (TP) if an HT-injected IC is labeled as compromised by the proposed framework and false negative (FN) if an HT-injected IC is labeled as safe. Alternatively, if an HT-free IC is labeled as safe, the prediction is considered true negative (TN). Finally, the prediction is a FP if an HT-free IC is labeled as compromised.

All the TP, FP, TF, and FN predictions are counted for all the TRIT-TC ICs, and the precision (P) and recall (R) performance metrics are calculated as follows $P = \frac{\text{TP}}{\text{TP+FP}}$, $R = \frac{\text{TP}}{\text{TP+FN}}$.

Note that the recall metric measures the ability of the system to identify HT-infected ICs correctly. While recall is the most important HT-detection metric, high precision helps to minimize the time required for human engineers to confirm the HT presence.

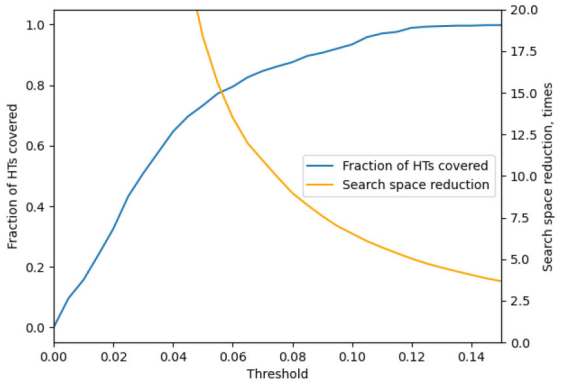

Fig. 10. Reduction of search space complexity and a fraction of covered HTs as a function of the radius threshold value (see Section III-A). The blue curve represents the fraction of HT embeddings preserved in the reduced search space. The orange curve represents the search space reduction rate.

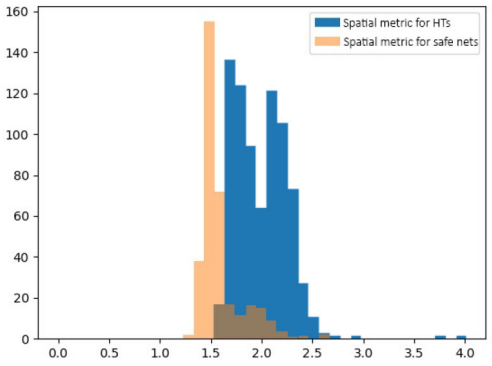

Fig. 11. Distribution of the spatial metric scores in all the combinational TRIT-TC IC for safe and HT net embeddings. The metric values for all the available HTs gathered from different combinational HT benchmarks ICs are shown in blue. These values are generally greater than the metric values of safe nets' embeddings, yet cannot be efficiently distinguished. The intersection between safe and HT histograms is shown in brown, illustrating the uncertainty of distinguishing between the two classes in most of the metric value range.

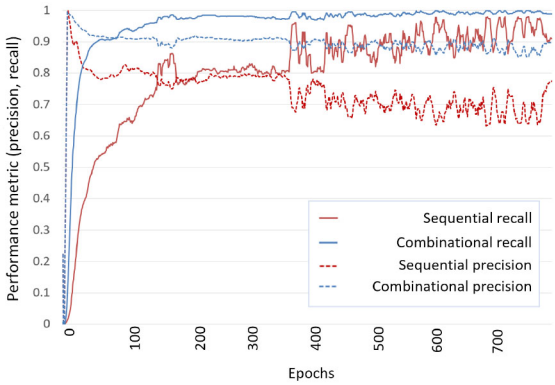

Fig. 12. HT detection training with the proposed supervised graph embedding convolution.

E. HT Detection Performance and Evaluation

To evaluate the proposed search space reduction policy, all the TRIT-TC HT ICs are embedded, and distances between the HT embedding centroids and all major embedding cluster centroids are measured. Recall that major IC embedding clusters represent various functional blocks within the IC, as determined by the OPTICS clustering algorithm. Embedding clouds of the TRIT-TC ICs follow one of the two different configurations—a monomodal distribution with a single major cluster of vectors, and a multimodal distribution with multiple major clusters. The histograms, as shown in Fig. 4, demonstrate the distance distribution for different types of HT for both configurations. The distance is illustrated with a blue bar if there is only one major cluster or if the HT embeddings are near the largest cluster. The distance is illustrated with an orange bar if the HT embeddings are closer to a smaller nonmajor (secondary) cluster in the multimodal embedding distribution.

As seen from the data illustrated in Fig. 10, a lower-radius threshold value reduces the search complexity by removing numerous safe embeddings from the search space. However, some of the HT embeddings can be lost with aggressive space reduction. The radius threshold value is therefore tuned to efficiently reduce the search space and yet conserve the essential information about the HT nets.

Since the histogram of distances between HT embeddings and the closest cluster centroids is a bell-shaped curve, we assume that the interval with a radius of 3σ (where $\sigma = 0.05$ is the standard deviation of empirical data rounded to two decimal places) covers 99.7% of unseen HTs. Therefore, a radius threshold value of 0.15 is considered to preserve 99.7% of the HT-related information while reducing the number of embeddings by 3.5 times, as illustrated in Fig. 10. Note that to label the overall IC as a compromised, zero-tolerance detection policy requires the identification of at least one HT-related gate-level hardware component. Since practical HT circuits comprise multiple gates, no performance reduction (in terms of detection accuracy) is observed with the proposed search space reduction.

The policy for detecting suspicious nets based on the spatial pattern score (see Section III-A) is also evaluated based on TRIT-TC ICs to determine the score threshold for HT detection. While the spatial patterns illustrated in Fig. 6 are visually apparent in some combinational HTs, other combinational HTs, along with most sequential HTs do not follow this spatial pattern. To demonstrate the distribution of the spatial pattern score for safe and HT net embeddings in all the combinational TRIT-TC ICs is illustrated in Fig. 11, exhibiting a quite clear distinction between safe and compromised nets and typical results for a human-engineered metric. While embeddings of HT-compromised nets overall exhibit a higher value of the proposed spatial metric and a threshold for high-HT detection accuracy can be determined, such a threshold cannot be used for developing a reliable high-recall HT prediction model.

To overcome the typical limitations of human-engineered metrics, robust HT detection is required to learn multiple spatial features from a training set autonomously. Thus, a supervised ML model, as described in Section III-B, is preferred. The training performance of the proposed supervised system is shown in Fig. 12.

The HT detection precision and recall performance is evaluated for all the combinational and sequential HTs from the TRIT-TC benchmark suite as a function of the number

TABLE II
HT-COMPROMISED BENCHMARKED ICS WHICH ARE UNDETECTED BY MOST OF THE TWELVE TRAINED MODEL INSTANCES

s1423_T002.v	s15850_T001.v	s15850_T019.v
s1423_T003.v	s15850_T007.v	s35932_T013.v
s1423_T011.v	s15850_T012.v	s35932_T014.v
s1423_T016.v	s15850_T015.v	s35932_T018.v

of training epochs. While the recall significantly increases, some decrease in precision metric is noticeable during the second half of the training process. This is due to the employed zero-tolerance detection policy, which prioritizes low-FN predictions (i.e., HT-compromised ICs should not be missed) at the expense of increased FP predictions (some safe ICs might be labeled as malicious). In order to evaluate the ability of the proposed ML model to generalize patterns and accurately classify new data points (i.e., high-model stability and low-model variance), twelve model instances were trained on distinct training datasets using a k-fold approach. These datasets were generated by running the embedded model with different random seeds. A robust, low-variance model is expected to exhibit comparable performance metrics with all the model instances and demonstrate a consistent difference between predicted and actual values (bias).

The number of HTs detected with each of the twelve model instances was tallied for all the HT-compromised TRIT-TC ICs, as depicted in Fig. 13. Majority of the 370 HTs are detected with high accuracy across all model instances, whereas certain HT benchmarks are frequently undetected in most training cases, resulting in a stable, systematic bias. Note that upon manual examination, the outlying benchmark ICs (as listed in Table II) do significantly differ from others. Also note, that none of these outlying ICs is detectable with the human-engineered spatial pattern metric.

To evaluate the capacity of the system to generalize HT properties, the model has been trained multiple times on datasets of various sizes and evaluated using the precision metric to assess the proposed system's ability to generalize properties of HTs. To construct the individual training subsets, combinational and sequential HT data has been randomly sampled (70%, 80%, 90%, and 95%), and evaluated. If the model can effectively generalize HT properties, little to no performance variation based on the training set size is expected. Model saturation rate changes with various training set sizes, resulting in performance variations. To mitigate the performance variations, the performance of the individual models as trained on sampled data is measured after various numbers of training epochs within the flat precision region (e.g., as exhibited between epochs 200 and 350 in Fig. 12). The model precision of the individual training runs with various training random seeds and various numbers of epochs exhibits no dependence on training set size. Please note that the precision scores in this experiment are computed per net, while the overall system performance shown in Table III is aggregated across the whole HT subcircuit in the corresponding IC. These results support the system's ability to generalize HT properties and detect previously unseen threats.

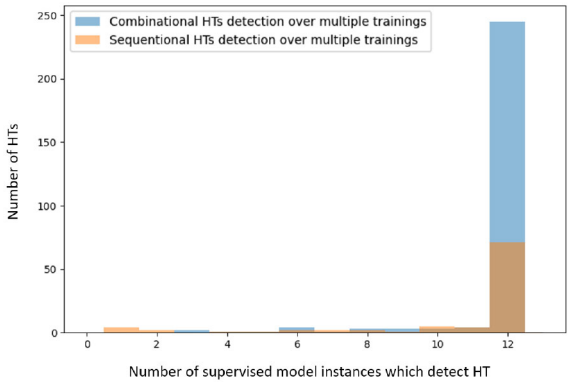

Fig. 13. Distribution of HTs based on the detectability with various numbers of trained model instances total of twelve trained model instances was trained with k-fold approach. The majority of HT benchmarks can be detected with all the model instances.

Performance of the existing state-of-the-art HT detection solutions is listed in Table III. To evaluate the scalability of the HT detection methods with IC size, time and memory complexity is listed as a function of the number of netlist components, n. If offline training and online inference are required, the complexity is given for either inference or a single training epoch with a training batch of minimum size (whichever is more complex).

The quantitative performance of the proposed framework is on par with modern state-of-the-art academic solutions. However, the advantages of the proposed framework as compared with the prior art extend beyond the detection accuracy. To the best of the authors' knowledge, the proposed framework is the first HT detector that simultaneously demonstrates all the required characteristics, as explained in Section II.

F. Runtime and Asymptotic Time and Memory Complexity

The proposed model is trained and evaluated using the NVidia GTX1080 GPU. The graph embedding stage is implemented using the BigGraph framework. This framework is proven effective for graphs of arbitrary size with no limitation on utilized hardware or available memory. The feature extraction and supervised model inference stages are executed in less than 0.1 s for each IC in the benchmark suite. The time complexity of the proposed algorithms for feature extraction and supervised classification in sparse design graphs is $O(nF)$, where n is the number of nets in the design, and F is the size of the reception field. The time complexity of the inference is $O(F)$ and is independent of the design size. The memory complexity of both feature extraction and model inference is $O(F)$, since each net is processed independently and consequently. Note that the inference process can be batched to increase the inference throughput at the cost of higher-GPU memory consumption in practical systems.

Another important runtime factor is the memory complexity of a training epoch with minimum batch size. Models cannot be efficiently trained if the minimum size batch (usually a batch of one sample) cannot fit into GPU memory. For example, a model in which training memory complexity is a function of the input size [e.g., $O(n)$], is by default limited to

TABLE III
PERFORMANCE OF STATE-OF-THE-ART HT DETECTION APPROACHES

	Detection level	Recall			Precision			Time complexity		Memory complexity	
		Combinational	Sequential	Overall	Combinational	Sequential	Overall	Worst case	Expected	Worst case	Expected
ISCAS'23 [21]	Gate	N/A	N/A	0.64	N/A	N/A	<0.7	O(n)	O(n)	O(1)	O(1)
HOST'21 [20]	Gate	0.95	1	N/A	0.9	0.92	N/A	O(n)	O(n)	O(n)	O(n)
TCAD'21 [18]	RTL, Gate	N/A	N/A	0.84	N/A	N/A	0.91	O(n)	O(n)	O(n)	O(n)
AsianHOST'18 [16]	RTL	N/A	N/A	0.87	N/A	N/A	0.85	$O(n^2)$	$O(n^2)$	O(1)	O(1)
ICCIT'19 [22]	RTL	0.98	0	N/A	0.98	0	N/A	$O(n^3)$	O(n)	O(n)	O(n)
ISCAS'19 [15]	RTL	N/A	N/A	1	N/A	N/A	0.92	O(n)	O(n)	O(1)	O(1)
ISCAS'17 [29]	Gate	N/A	N/A	0.93	N/A	N/A	1	O(n)	O(n)	O(1)	O(1)
IOLTS'21 [11]	Gate	N/A	N/A	0.64	N/A	N/A	1	O(n)	O(n)	O(1)	O(1)
This work	Gate	**0.99**	**0.91**	**0.96**	**0.9**	**0.77**	**0.86**	**O(n)**	**O(n)**	**O(1)**	**O(1)**

smaller inputs (which can be fitted into GPU memory), and cannot be practically trained with large inputs. Consequently, HT detection methods that rely on existing GCN approaches exhibit limited memory scalability, as shown in Table III with [18] and [20]. Alternatively, the proposed convolution model uses a fixed size input per net, exhibiting input-independent memory complexity of $O(F)$. Thus, the proposed method can be used for HT detection in ICs of any size.

V. CONCLUSION

A scalable reference-free framework is introduced in this article for netwise gate-level detection of existing and unknown HTs. With the proposed approach, an embedding cloud is generated from a graph-based netlist representation. The embeddings of the individual IC nets are convolved in a supervised manner to detect anomalies within the embedding cloud. By automatically extracting relevant connectivity information from the netlist and learning the underlying IC behavior, the proposed model can effectively identify previously unseen HTs. The constant space complexity and linear time complexity of the underlying algorithms make the proposed approach effective for HT detection in ICs of any size. The performance of the proposed framework was experimentally demonstrated using the TrustHub TRIT-TC benchmark suite, which includes combinational and sequential ICs of various sizes. For gate-level detection with a zero-tolerance detection policy, the proposed method is shown to correctly identify HTs with a high recall of 96% and a precision of 86%. The individual HT-compromised nets are highlighted within less than 0.1 s across all tested ICs (independently of their individual sizes).

REFERENCES

[1] J. Rajendran, V. Vedula, and R. Karri, "Detecting malicious modifications of data in third-party intellectual property cores," in *Proc. 52nd Annu. Design Autom. Conf.*, 2015, pp. 1–6.

[2] J. Rajendran, A. M. Dhandayuthapany, V. Vedula, and R. Karri, "Formal security verification of third party intellectual property cores for information leakage," in *Proc. 29th Int. Conf. VLSI Design 15th Int. Conf. Embed. Syst. (VLSID)*, 2016, pp. 547–552.

[3] O. Aramoon and G. Qu, "Impacts of machine learning on counterfeit IC detection and avoidance techniques," in *Proc. 21st Int. Symp. Qual. Electron. Design (ISQED)*, 2020, pp. 352–357.

[4] L. Azriel, J. Speith, N. Albartus, R. Ginosar, A. Mendelson, and C. Paar, "A survey of algorithmic methods in IC reverse engineering," *J. Cryptogr. Eng.*, vol. 11, no. 3, pp. 299–315, 2021.

[5] S. Saha, R. S. Chakraborty, S. S. Nuthakki, Anshul, and D. Mukhopadhyay, "Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability," in *Proc. 17th Int. Workshop, Cryptogr. Hardw. Embed. Syst. (CHES)*, 2015, pp. 577–596.

[6] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using boolean functional analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 697–708.

[7] 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. 10796–10826, 2020.

[8] D. Utyamishev and I. Partin-Vaisband, "Real-time detection of power analysis attacks by machine learning of power supply variations on-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 1, pp. 45–55, Jan. 2020.

[9] F. Kenarangi and I. Partin-Vaisband, "Security network on-chip for mitigating side-channel attacks," in *Proc. ACM/IEEE Int. Workshop Syst. Level Interconnect Predict. (SLIP)*, 2019, pp. 1–6.

[10] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, 2017, pp. 227–232.

[11] T. Kurihara and N. Togawa, "Hardware-trojan classification based on the structure of trigger circuits utilizing random forests," in *Proc. IEEE 27th Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, 2021, pp. 1–4.

[12] K. Hasegawa, S. Hidano, K. Nozawa, S. Kiyomoto, and N. Togawa, "R-HTDetector: Robust hardware-trojan detection based on adversarial training," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 333–345, Feb. 2023.

[13] L. Piccolboni, A. Menon, and G. Pravadelli, "Efficient control-flow subgraph matching for detecting hardware trojans in RTL models," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, pp. 1–19, 2017.

[14] M. Fyrbiak, S. Wallat, S. Reinhard, N. Bissantz, and C. Paar, "Graph similarity and its applications to hardware security," *IEEE Trans. Comput.*, vol. 69, no. 4, pp. 505–519, Apr. 2020.

[15] T. Han, Y. Wang, and P. Liu, "Hardware trojans detection at register transfer level based on machine learning," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2019, pp. 1–5.

[16] F. Zareen and R. Karam, "Detecting RTL trojans using artificial immune systems and high level Behavior classification," in *Proc. Asian Hardw. Orient. Secur. Trust Symp. (AsianHOST)*, 2018, pp. 68–73.

[17] F. Demrozi, R. Zucchelli, and G. Pravadelli, "Exploiting sub-graph isomorphism and probabilistic neural networks for the detection of hardware trojans at RTL," in *Proc. IEEE Int. High Level Design Valid. Test Workshop (HLDVT)*, 2017, pp. 67–73.

[18] R. Yasaei, L. Chen, S.-Y. Yu, and M. A. Al Faruque, "Hardware trojan detection using graph neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, May 26, 2022, doi: 10.1109/TCAD.2022.3178355.

[19] R. Yasaei, S.-Y. Yu, and M. A. Al Faruque, "GNN4TJ: Graph neural networks for hardware trojan detection at register transfer level," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2021, pp. 1504–1509.
[20] N. Muralidhar, A. Zubair, N. Weidler, R. Gerdes, and N. Ramakrishnan, "Contrastive graph convolutional networks for hardware trojan detection in third party IP cores," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2021, pp. 181–191.
[21] H. Lashen, L. Alrahis, J. Knechtel, and O. Sinanoglu, "TrojanSAINT: Gate-level netlist sampling-based inductive learning for hardware trojan detection," 2023, *arXiv:2301.11804*.
[22] S. A. Islam, F. I. Mime, S. M. Asaduzzaman, and F. Islam, "Socio-network analysis of RTL designs for hardware trojan localization," in *Proc. 22nd Int. Conf. Comput. Inf. Technol. (ICCIT)*, 2019, pp. 1–6.
[23] D. Utyamishev and I. Partin-Vaisband, "Knowledge graph embedding and visualization for pre-silicon detection of hardware trojans," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2022, pp. 180–184.
[24] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 2071–2080.
[25] A. Lerer et al., "Pytorch-BigGraph: A large scale graph embedding system," in *Proc. Mach. Learn. Syst.*, 2019, pp. 120–131.
[26] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, 1999.
[27] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1–10.
[28] L. Van Der Maaten and G. Hinton, "Visualizing data using T-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.
[29] 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 *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2017, pp. 1–4.

Dmitry Utyamishev (Student Member, IEEE) received the B.Sc. degree in computer engineering from National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, Russia, in 2014. and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Chicago, Chicago, IL, USA, in 2022 and 2024, respectively.

His current research interests include robotics, AI, and high-scale design optimization.

Inna Partin-Vaisband (Senior Member, IEEE) received the B.Sc. degree in computer engineering and the M.Sc. degree in electrical engineering from the Technion-Israel Institute of Technology, Haifa, Israel, in 2006 and 2009, respectively, and the Ph.D. degree in electrical engineering from the University of Rochester, Rochester, NY, USA, in 2015.

She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL, USA. From 2003 to 2009, she held a variety of software and hardware research and development positions with Tower Semiconductor Ltd., Migdal Haemek, Israel; GConnect Ltd., Petah Tikva, Israel; and IBM Ltd., Haifa, Israel. Her primary interests include high-performance integrated circuits and VLSI system design. Her current research focused on innovation in the areas of AI hardware and hardware security. Yet, another primary focus is on distributed power delivery and locally intelligent power management that facilitates performance scalability in heterogeneous ultralarge-scale integrated systems. Special emphasis is placed on developing robust frameworks across levels of design abstraction for complex heterogeneous integrated systems.

Dr. Partin-Vaisband is an Associate Editor of the *Microelectronics Journal* and has served for the Technical Program and Organization Committees of various conferences.