2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

HTtext: A TextCNN-based pre-silicon detection for hardware Trojans

Yi Xu[†], Zhenyi Chen[‡], Binhong Huang[†], Ximeng Liu[†§], Chen Dong[†¶*]
[†]College of Computer and Data Science, Fuzhou University, Fuzhou, China
[‡]Department of Electrical Engineering, University of South Floride, Tampa, USA
[§]Key Lab of Information Security of Network Systems (Fuzhou University), Fujian Province
[¶]Fujian Key Laboratory of Network Computing and Intelligent Information Processing (Fuzhou University)
Email: xuyilaser@foxmail.com, zhenyichen@usf.edu, ahaiamhuang@foxmail.com, snbnix@gmail.com, *dongchen@fzu.edu.cn

***Abstract*—With the advent of the intelligence era, the usage and investment for chips have risen year by year, causing the demand for chip security. Machine learning (ML) analysis has made progress as a pre-silicon hardware Trojan (HT) detection technology. Whereas the performance of existing methods almost relies on the accuracy of multi-feature representation, moreover, it is difficult to extract features manually and easily cause unstable classifier performance (namely uncertainty). In this paper, an automatic feature extraction detecting model is first proposed, named HTtext, which generates simple path sentences from chip netlists and employs TextCNN, a deep learning algorithm, to distinguish HT circuits. The pre-training for TextCNN only uses the automatic single-feature calculation to avoid the uncertainty problem. Additionally, the model can obtain non-repetitive HT component information expression, which satisfies the stable detection performance. To measure the efficiency and balance of the model, the paper proposes the concept of the Stability Efficiency Index (SEI). In the experimental results for the benchmark netlists, not only the average accuracy (ACC) in TextCNN is as high as 99.26%, but also its SEI value ranks first in all comparison classifiers, which proves that the proposed HTtext model has high stability in generality.**

***Index Terms*—hardware Trojan, chip security, deep learning, simple path sentences generating, chip netlist, TextCNN**

I. Introduction

With the advent of the fourth industrial revolution (industry 4.0), wireless networks [1] and the 6th generation mobile networks, the concept of artificial intelligence (AI) has continued to develop and gradually penetrates chip applications. The Semiconductor Industry Association (SIA) report in the USA on April 2021 indicated that the world's investment demand for chips will increase to 3 trillion US dollars in the next decade [2].

Extensive application space and broad development prospects put forward higher requirements for ensuring chip security [3], since once an attacker launches an attack by the HT [4], it will lead to catastrophic consequences, such as leakage of information, denial of service (DoS), degrade performance [5] and even destruction of the entire system. This adverse impact can be extended to other areas where integrated circuit (IC) is used. Like software, to counter the high concealment and unpredictable destructiveness of HT, high efficiency and reliable HT identification technology is the foreseeable only effective way.

Due to intellectual property and cost reasons, there are many third parties involved in the chip design and manufacturing cycle, which causes serious security risks for IC. Nowadays, there are various technical methods for HT detection, which can be divided into four categories. (1) *Side-channel detection technology*, this technology utilizes side channels [6] such as current, voltage, power, and temperature to detect redundant information in comparison with the golden IC to find HT. (2) *Reverse engineering*, scanning it layer by layer and destructing the chip, the inconsistencies in the structure are compared with the golden chip on the physical image [7]. (3) *Logic testing*, inputting a series of test vectors in the IC simulation stage to trigger the HT attack phenomenon [8]. The three types mentioned above are easily affected by external factors, need the golden chip, and just suitable for smaller scale IC. For very large scale integrated (VLSI), the ML-based method is a good choice. (4) *Static detection combined with ML*, which is an emerging technology that does not rely on golden chip [9] or professional equipment, and can adapt to deal with VLSIs.

Whereas, the design stage has more vulnerabilities than the manufacturing stage [10], as the EDA (Electronic Design Automation) tools, IP (Intellectual Property) core construction, and netlist files are easily implant HT by malicious internal employees or designers. In view of the advantages of ML, static detection combined with ML is a promising approach for detecting HT of VLSIs. Dong et al. [11] screened out 20 valuable features from the extracted 51 circuit features and used BP neural network algorithm and support vector machine (SVM) to detect HT netlists for RS type and S type. Tatsuki et al. [12] only calculated the HT results of the random forest (RF) algorithm for 11 netlist features, and the RS type netlist date has better results.

The detection based on traditional ML [13] (e.g. SVM, decision tree algorithms) is oriented towards feature construction and screening, and mostly faces the difficulty of manual extraction and the performance relies on the representation and judgment of features. Traditional ML cannot effectively calculate more complex nonlinear data models [14], and these algorithms are difficult to handle diverse circuit information.

Deep learning (DL) cannot only deal with large-scale and multiple types of circuit data, but more significantly, the feature extraction is an automated process, avoiding the te-

978-0-7381-2646-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00022

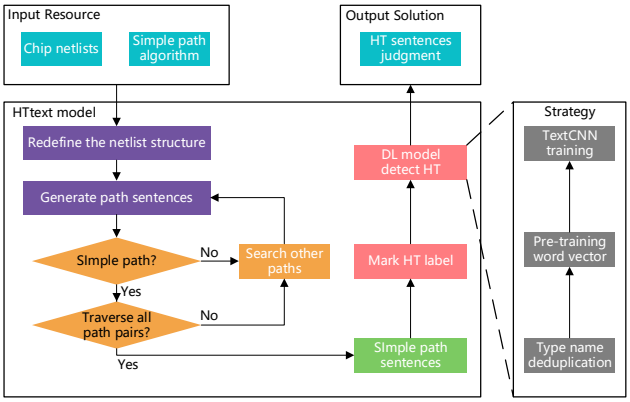

Fig. 1. Overall flow for HTtext model detection.

diousness and uncertainty of manual extraction [15]. This paper proposed a pre-silicon detection based on DL, HTtext in short, overcomes the shortcomings of the above-mentioned technologies and even traditional ML algorithms. Since TextCNN [16] is a one-dimensional convolution kernel training in units of words, it is more suitable for text recognition than ordinary CNN. HTtext employs word2vec [17] automatic feature extraction and combines with TextCNN to identify HT netlist, the contributions are as follows:

- Proposing the first work that is the sentences-based pre-silicon DL technology for HT detection. Eliminating the dependence on the golden chip comparison and reduce the detection cost.
- Generating simple path sentences amplifies the HT information without repeating it, thereby overcoming the rarity for HT in the circuit structure.
- Establishing automatic single-feature HT detection for circuit components avoids the uncertainty of manually extracting features.
- Proposing the SEI indicator to measure the high performance and stability of the detection model, and the average ACC has an achieve of 99.26% on the HTtext model.

The rest of the paper is organized as follows: Section II gives the motivation and overview for the HTtext model. Section III describes the simple path sentences generating. Section IV explains the vocabulary training for type names. Section V declares the mechanism and parameters of TextCNN to detect HT. Section VI analyzes the performance of HTtext. Finally, Section VII is the conclusion.

II. Motivation and Details for the HTtext Model

A. Problem description

In chip designing stage, the attackers implant HTs components $H = \{h_1, h_2, h_3, \cdots, h_k\}$ into a series of chip netlists $R = \{r_1, r_2, r_3, \cdots, r_m\}$. After the rare condition $C = \{c_1, c_2, c_3, \cdots, c_p\}$ is activated, the path $P = \{p_1, p_2, p_3, \cdots, p_v\}$ containing HTs will cause damage such as signal tampering through signal transmission.

TABLE I
Symbols and Notations

Symbols	*Definitions*
n_i	ith component in component set N
s_i	ith wire in wire set S
$f_{spath}/f_{spathHT}$	the simple path/ the HT simple path
T_{spath}	the simple path sentence
SE_{HT}	the models SEI for sample recognition
$SE_{overall}$	the models SEI for all ML indicator
X_{ii}	the set of in-wire in the component n_i
X_{io}	the set of out-wire in the component n_i
X_{iport}	the wire set of input ports
X_{oport}	the wire set of output ports
f_s	the signal transfer function in the circuit
f_{sp}	a simple path function
S_{path}	the node-set for the beginning component
F_{path}	the node-set for the ending component

Where m and k respectively mean the number of netlists and components, and p and v are the number of conditions and paths, respectively.

For TextCNN to effectively detect the HT netlist during the chip design stage, the detection problem can be defined as

- ***Input:*** **(i)** The chip netlist files contain with component set $N = \{n_1, n_2, n_3, \cdots, n_l\}$ and wire set $S = \{s_1, s_2, s_3, \cdots, s_t\}$, where l and t respectively represent the total number of components and the total number of wires. **(ii)** The algorithm that can search the simple path $f_{spath} : s_i \rightarrow s_j$ from the input port wire s_i to the output port wire s_j.
- ***Output:*** A detection result for judging whether the simple path sentence T_{spath} is an HT simple path $f_{spathHT}$.
- ***Objective:*** Maximize SE_{HT} and $SE_{overall}$.

Table I presents the symbols and notations that are used throughout this paper.

B. HTtext model overall detection description

To solve the problem in Subsection A, the paper proposes the HTtext model as shown in Fig. 1. Redefining the netlist structure according to the netlist code to extract component and wire information. Furthermore, the existing simple path algorithm generates sentences and determines whether it is a simple path by the relationship between the components connected to the same wire. Eventually, the generated simple path sentences are labeled with HT, and the HT path judgment is completed through word vector pre-training and TextCNN learning. The detection system is the HTtext model that can be used to solve the HT netlist problem.

III. Netlist Preprocessing and Sentence Generating

The preprocessing work is responsible for transforming the netlist code into a data set that can be recognized by the DL model. Generating sentences is the core means to realize the transformation process.

A. Redefinition for netlist structure

The meaning of redefining the netlist is to divide the netlist structure reasonably and filter out the information materials needed to generate sentences.

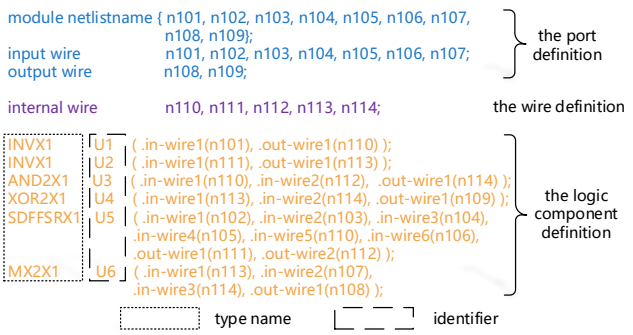

Fig. 2. Netlist code structure division.

Fig. 2 shows the netlist code file written in Verilog language. The entire code segment is divided into three parts: the port definition part, the wire definition part, the logic component definition part.

1) the port definition part: This part lists the wires that are considered to be connected to the input and output ports.

2) the wire definition part: The content indicates the identifier information of wires in the circuit except for the port wires included in the first part, which they called the internal wires.

3) the logic component definition part: It not only lists the components required to form a complete circuit but also follows them with in-wire and out-wire information.

Each component receives the signal from the in-wire and outputs the signal processed by the logic operation from the out-wire. This phenomenon indicates that the signal between components follows the rule that the wires with the same identifier can be transmitted. Also, all port wires and internal wires are included in the categories of in-wire and out-wire.

B. Simple path sentences generation

Generating text information from the netlist can provide the materials needed to detect HT through semantic understanding, and the text form is the simple path sentences. The TextCNN model learns these sentences to understand the circuit layout and find HTs.

1) Establishing for circuit signal transmission law

The proposed detection model determines the word order expression in sentences according to the way of signal transmission and forms a recognizable text data set from the redefined netlist structure according to the proposed law.

Accordingly, the component set in Section II is $N = \{n_1, n_2, n_3, \cdots, n_l\}$, and the elements in the set can correspond to each component identifier in the third part of the code. Also, the set S represents all the wire elements in the netlist, which can correspond to each internal wire in the second part and the port wires of the first part in Fig. 2. For component n_i and component n_j $(1 \le i \le l, 1 \le j \le l, i \ne j)$, the wire information it contains can be expressed as $n_i\,[X_{ii}, X_{io}]$ and $n_j\,[X_{ji}, X_{jo}]$ $(X_{ii}, X_{io}, X_{ji}, X_{jo} \subset S)$, respectively. Among them, X_{ii} and X_{io} are respectively represented as the set of in-wire and out-wire in the component n_i. For n_j, X_{ji} and X_{jo} also correspond to the concepts for in-wire and out-wire respectively. Then the signal transmission law can be expressed as the following formula:

$$X_{io} \cap X_{ji} \neq \varnothing \Rightarrow f_s : n_i \rightarrow n_j \tag{1}$$

The formula (1) is interpreted as if an out-wire identifier in n_i is consistent with an in-wire identifier in n_j, it means that the circuit signal can be passed from component n_i to component n_j. In the above formula, f_s is defined as the signal transfer function in the circuit, and this function is the core embodiment for the signal transfer law, which shows the basic characteristics of the netlist.

2) Filtering port words

The purpose of determining the port word is to find the start component and the end component on a path pair. These components set clear boundaries for the path sentence in the form of words.

For n_i, let X_{iport} and $X_{oport}(X_{iport}, X_{oport} \subset S)$ be the wire set of input and output ports, then the beginning word (the first component word in the sentence) and the ending word (the last component word in the sentence) of each path sentence can be obtained by the following formulas:

$$X_{ii} \cap X_{iport} \neq \varnothing \Rightarrow n_i \in S_{path} \tag{2}$$

$$X_{io} \cap X_{oport} \neq \varnothing \Rightarrow n_i \in F_{path} \tag{3}$$

S_{path} and F_{path} in formulas (2) and (3) are the node-set for the beginning component and the ending component, respectively. For formula (2), in simple terms, the in-wire set and the port wire set in n_i have the same wire identifier, then consider the component n_i as the starting word for a path sentence; In formula (3), the out-wire set in n_i has the same wire identifier as the port wire set, and the component n_i is considered as the last word for a path sentence.

The essence of the starting word and ending word corresponds to the components connected to the input and output ports in the netlist circuit. Only when these components are used as the starting and ending points to derive the path sentence information contained therein can it be fully reflected a panoramic view of signal transmission throughout the circuit.

3) Searching paths and judging simple paths

The sentence information can be filled with the search path, and during the filling process, it is ensured that the sentence is in strict compliance with the simple path format.

After the port words are determined, the above formula (1) can be used to complete the remaining path sentence content, which can be obtained according to the following formula:

$$Path: \ f_s^v(n_i) = n_j, where\ n_i \neq n_j \tag{4}$$

Where v represents the number of iterations for the function f_s. Then according to the formula (4), the component n_i is the starting point, and the signal transfer function is iterated for v times to form a path leading to n_j as the ending point, and the starting point and the ending point are not the same components (namely they are not the same node). Based on

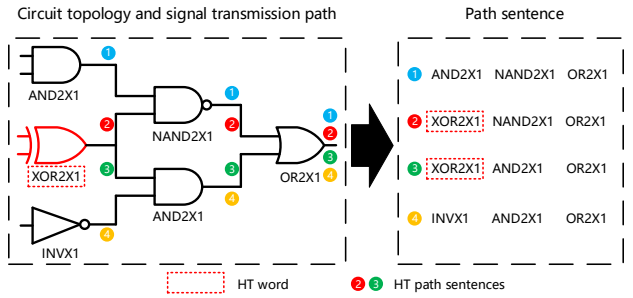

Fig. 3. Converting circuit topology into path sentences and marking HT word.

formula (4), it is necessary to find sentence constraints for constructing a simple path, as shown in formula (5):

$$\forall a, \forall b \left(f_s^a\left(n_i\right) \neq f_s^b\left(n_i\right)\right) \Rightarrow f_s^v = f_{sp} \\ where\ 0 \leq a \leq v, 0 \leq b \leq v, a \neq b \tag{5}$$

Where f_{sp} is expressed as a simple path function. According to formula (5), any node after a times of signal transmission is not the same as any node after b times of signal transmission except a times, and the path formed after such transmission is a simple path. That is to say, each node in the searched path (corresponding to the word in the path sentence) is different from any other node on the path under the constraint of formula (5), which shows that the path can be expressed as a simple path sentence.

Fig. 3 manifests that under the effect of formulas (4) and (5), the circuit signal is only transmitted on the icons of the same number to generate a simple path. The components on each such path are recorded to the sentence in turn according to their identifiers. If there is an HT component on the simple path (i.e. the red component in Fig. 3), then the corresponding generated text is an HT sentence.

C. Reasons for relying on the simple paths

The reason why the detection framework proposed in the paper is to generate text sentences based on simple paths is that the loop feature information does not have an evaluation advantage for the recognition of HT circuits in the past reference [11], [18]. In [18], although some loop features play a significant role in the HT classification, the ratio is low compared to the number of all loop features.

At the same time, the experimental results also pointed out that the excellent recognition effect was still obtained under the lack of loop information, and also proved that it is feasible to simplify the sentence expression by outputting only the information for simple paths.

D. The overall flow for generating simple path sentences

Based on the above formulas and rules, the paper sorts out a complete and detailed text preprocessing process and displays it in **Algorithm 1**.

Firstly, the detection framework generates all component identifiers and all wire identifiers based on the given netlist. Furthermore, adding in-wire and out-wire information to each

Algorithm 1: Generating simple path sentences

Input: The set N of components, the set S of wires in the netlist
Output: All of the simple path sentences for expressing the netlist

1 Initialize the in-wire and out-wire sets X_{ii}, X_{io} and X_{ji}, X_{jo} for n_i, n_j; extract the wire sets X_{iport}, X_{oport} from the port definition part.
2 **for** *each* $n_i \in N$ **do**
3 $\quad n_i\left[X_{ii}, X_{io}\right] \leftarrow$ get in-wire and out-wire information from S;
4 **end**
5 **for** *each* $s_i \in S$ **do**
6 $\quad X_{iport} \leftarrow$ get input port wires from S;
7 $\quad X_{oport} \leftarrow$ get output port wires from S;
8 **end**
9 **for** *each* $n_i\left[X_{ii}, X_{io}\right]$ **do**
10 $\quad$ **if** $X_{ii} \cap X_{iport} \neq \varnothing$ **then**
11 $\quad\quad n_i \in S_{path}$;
12 $\quad$ **end**
13 $\quad$ **if** $X_{io} \cap X_{oport} \neq \varnothing$ **then**
14 $\quad\quad n_i \in F_{path}$;
15 $\quad$ **end**
16 **end**
17 **for** *each* $n_i \in S_{path}$ **do**
18 $\quad$ **for** *each* $n_j \in F_{path}\ (j \neq i)$ **do**
19 $\quad\quad$ **if** $f_s^v\left(n_i\right) = n_j$ *and* $f_s^v = f_{sp}$ **then**
20 $\quad\quad\quad$ SimplePath $\leftarrow f_{sp}\left(n_i\right) = n_j$;
21 $\quad\quad$ **end**
22 $\quad\quad$ **if** $ComputingTime > 1min$ *or* $SumSimplePath\left(n_i, n_j\right) = 100000$ **then**
23 $\quad\quad\quad$ *continue* (jump to the next **for** loop);
24 $\quad\quad$ **end**
25 $\quad$ **end**
26 **end**
27 **for** *each SimplePath* **do**
28 $\quad$ SimplePathsentence $\leftarrow$ Converting identifiers to type names for SimplePath;
29 **end**
30 **return** SimplePathsentence

component provides a basis for the word order of the generated sentences (line 1-3). Moreover, putting all the identifiers belonging to the port wires in their respective input set and output set (line 4-7). The components that meet formulas (2) and (3) are selected and determined as the beginning word and the ending word in the path sentence, respectively (line 8-15).

Assuming that the netlist finally generates i beginning words and j ending words, then $i \times j$ path sentence expressions will be produced, and each expression contains a plurality of sentences. From the netlist mapping to the topological graph, there are multiple paths to a pair of determined beginning points and ending points, so the expression for these paths into sentences is also diverse. Calculating the expression for each path sentence to satisfy that the words contained in each sentence are not repeated (i.e. simple path sentence). In addition, when the calculation time exceeds 1 minute but does not get a complete sentence (timeout sentence) or the total number of sentences in a path pair reaches 100,000, the algorithm stops the operation on this path pair and turns to calculate the next path pair (line 16-25).

Setting the upper limit of 100,000 sentences is based on the need to collect enough sentence data sets. Abandoning timeout sentences is to consider only focusing on relatively simple topological expressions. Words originally in the form of identifiers in all obtained sentences are transformed into corresponding type names (line 26-29).

The reason for processing word expression conversion is

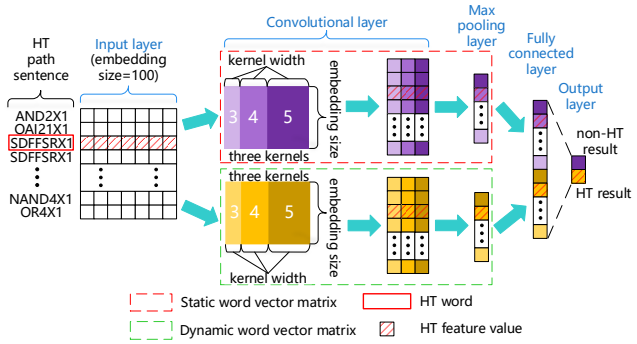

Fig. 4. TextCNN structure for detecting HT path sentences.

that identifiers are too diversified and irregular in naming for the vocabulary. If the identifiers are retained, vocabulary will be too large, which is not conducive to identifying HT. The type name has a unified naming form for all circuit components, which is conducive to the collation of the vocabulary and the training of word vectors. Finally, the exported simple path sentence can be entered into TextCNN to complete the model training.

The time complexity is $O\left((n_l + c) \times (e + 1)\right)$, where c and e are the numbers of simple loops and the number of edges connected between components.

IV. Type Names Pre-Training

The pre-training for the type name is to let the word vector record the relative relationship of the circuit components in the position. To better reflect the correlation between words from the word vectors, word2vec for unsupervised learning is used to calculate the clustering relationship among all the words in the vocabulary after deduplication.

A. Building the word vector matrix

The detection method in this paper uses the trained word vector to construct the static word vector matrix in the next stage, and pre-training needs to complete a set of mature word vector expressions for the matrix. The processing for TextCNN combined with the static word vector matrix is conducive to the faster convergence of the model, and even to a certain extent, the recognition accuracy can be improved, which is the reason why the paper adopts the pre-training mechanism.

B. HT information of word vectors in pre-training model

There are two models for word2vec's vocabulary training, namely the continuous bag-of-word model (CBOW) and the skip-gram [19].

The paper takes advantage of the latter model since its word vector undergoes more adjustments than the CBOW in prediction, and it has a greater possibility of obtaining a superior recognition effect. The formation for a mature word vector expression is measured by the following two formulas:

$$\log P\left(w_o \mid w_c\right) = \boldsymbol{u}_o^{\top} \boldsymbol{v}_c - \log\left(\sum_{i \in \nu} \exp\left(\boldsymbol{u}_i^{\top} \boldsymbol{v}_c\right)\right) \quad (6)$$

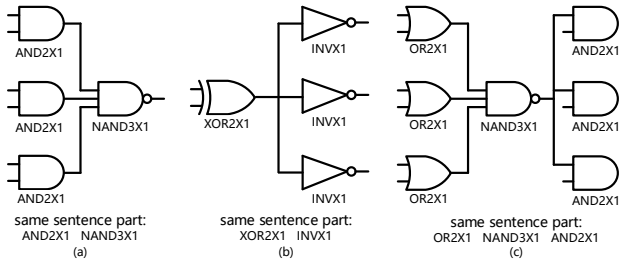

Fig. 5. Three cases of sentence repetition. (a) many-to-one (b) one-to-many (c) many-to-many

$$\frac{\partial \log P\left(w_o \mid w_c\right)}{\partial \boldsymbol{v}_c} = \boldsymbol{u}_o - \sum_{j \in \nu} \left(\frac{\exp\left(\boldsymbol{u}_j^{\top} \boldsymbol{v}_c\right)}{\sum_{i \in \nu} \exp\left(\boldsymbol{u}_i^{\top} \boldsymbol{v}_c\right)}\right) \quad (7)$$

In formula (6) [17], w_o and w_c represent surrounding words and a center word respectively. When the word as the HT component is w_c, the corresponding $\boldsymbol{v}_c$ is the word vector of the central word, the word vector $\boldsymbol{u}_i$ representing any word in the thesaurus, and the word vector $\boldsymbol{u}_o$ representing the surrounding component words are used to calculate the circuit layout probability.

When the word as the HT component is w_c, the corresponding $\boldsymbol{v}_c$ is the word vector of its central word, $\boldsymbol{u}_i$ represents any word vector in the vocabulary, and $\boldsymbol{u}_o$ represents the word vector for the surrounding component words. The above three word vector parameters are used to calculate the circuit layout probability. After the probability is normalized, $\boldsymbol{v}_c$ learns enough path sentences to record the distribution of components around the HT.

To make the word vector more accurately express the circuit layout to judge HT, the derivative for formula (6) becomes the gradient function, namely formula (7) [17]. The $\boldsymbol{u}_j$ in formula (7) makes the HT word that is the central word w_c become the word vector represented by the surrounding words and then combines the gradient descent algorithm to minimize the formula (6), and the finally exported $\boldsymbol{v}_c$ is the required word vector. The above process also has the same effect for learning circuit layout, when words represent non-HT components.

The above pre-trained word vector already contains the location information for the HT component in advance, which lays an optimized foundation for TextCNN to detect HT by further calculating the text features of simple path sentences.

C. Pre-training parameters

The range of selected surrounding words is 5. According to the size of the generated text data and the number of words in the vocabulary, choosing a word dimension of 100 to meet the efficiency.

V. TextCNN Attributes in HTtext Model

The construction for a reasonable TextCNN structure and efficient parameter selection are the key attributes that need to be considered in this model, and they play a significant role in the optimization of the model.

TABLE II
COMPLETE EXTRACTION RATE, LABEL STATISTICS, AND SAMPLING COVERAGE IN SEVEN NETLISTS

Benchmark	*Trojan-free sentences*	*Trojan sentences*	*Sampling rate*	*Path pairs complete rate*
RS232-T1000	5669588	932730	0.76%	174/180
RS232-T1100	6039452	562673	0.76%	174/180
RS232-T1200	5971413	600335	0.76%	168/180
RS232-T1300	6107276	1022651	0.70%	180/180
RS232-T1400	5973678	600327	0.76%	168/180
RS232-T1500	5439318	1163000	0.76%	174/180
RS232-T1600	5946976	1200526	0.70%	180/180
Total	41147710	6082242	0.74%	-

A. TextCNN construction

The frame construction for TextCNN refers to the [19], as shown in Fig. 4. All path sentences after text preprocessing are put into the input layer one by one, and TextCNN uses a one-dimensional convolution layer to calculate the feature of multiple convolution kernels for the HT words and non-HT words in the sentence.

The results obtained are put into the maximum pooling layer and the most valuable feature for judging HT (i.e. the maximum value feature) is selected, and then features concentrated in the fully connected layer. Finally, a linear function is used to synthesize multiple neurons in the fully connected layer to obtain the output of whether the sentence is an HT path.

The combination for static and dynamic word vector matrix can simultaneously consider the results of iterative non-update and update in TextCNN to optimize HT detection.

B. TextCNN parameters

1) sentence length setting: The paper considers the processing for truncating sentences (about 20% of the length is truncated) and the sentence length is set to 100. This approach reduces the model overhead while not affecting the detection rate.

2) convolution kernel width setting: Since it is a one-dimensional convolution layer, only three types of widths to convolution kernels are selected, which are 3, 4, and 5 respectively. There are two convolution kernels of each width and a total of six kernels. Two groups of three kernels with different widths are assigned to the static word vector matrix and the dynamic word vector matrix.

3) dropout setting: The dropout value in the fully connected layer is 0.5 by default to alleviate the overfitting.

VI. EXPERIMENT SETTING AND RESULTS

A. Equipment parameters and benchmark

The HT recognition framework for TextCNN proposed in the paper is implemented by the PyTorch code framework in Python language and tested on personal computers (PCs) such as the CPU of i5-7400HQ, the graphics card of GTX 1050Ti, and 16GB of memory. The seven benchmark netlists required for the experiment are derived from the public website Trust-Hub [20], the threat effect in these netlists is signal tampering.

TABLE III
THE RESULT OF TEXTCNN'S CLASSIFICATION FOR PATH SENTENCES

Benchmark	*TPR*	*TNR*	*PRE*	*ACC*	*F-measure*
RS232-T1000	99.97%	95.95%	96.11%	97.96%	98.00%
RS232-T1100	99.95%	100%	100%	99.98%	99.98%
RS232-T1200	99.98%	100%	100%	99.99%	99.99%
RS232-T1300	99.99%	97.11%	97.19%	98.55%	98.57%
RS232-T1400	99.98%	100%	100%	99.99%	99.99%
RS232-T1500	95.75%	100%	100%	98.37%	98.35%
RS232-T1600	99.98%	100%	100%	99.99%	99.99%
Average	99.51%	99.01%	99.04%	99.26%	99.27%

B. Vocabulary size

The number of words in the vocabulary in the pre-training stage is 15, which is the result of deduplication and cleaning of all the words in the netlist sentence.

C. Coverage of data set generating

In the netlist preprocessing stage, each netlist has 180 path pairs. Table II lists the number of non-timeout path pairs in each benchmark netlist. According to the overall statistics, only a few path pairs are not completely extracted, which can indicate that the generated sentences basically express the topology of the netlist.

In addition, the topological expression for these timeout path pairs may be supplemented in other completely extracted path pairs, since there is a certain probability that non-timeout path pairs will pass through the blank topology behind the timeout path pair.

D. Data set division and sampling

In the training model stage for TextCNN, the training set and the test set are divided according to the leave-one-out method. That is to say, the sentences generated from one netlist are used as the test set each time, and the sentences in the remaining netlists are regarded as the training set.

Since the collected netlist sentences are too large in number and to obtain more excellent experimental results, the balanced sampling for the data set is implemented. That is, each netlist is randomly selected 25,000 positive samples (with HT sentences) and 25,000 negative samples (with HT sentences), and a total of 50 groups of such sampling data are formed for all netlists. The determination for the number of samples is the result of the combination of the data sets and computational overhead required for neural network training.

According to Table II, the amount of data sampled by a single group (a total of 350,000 sentences) accounts for about 0.74% of all data.

From another aspect, the sentences generated within a single path pair have repetitiveness in expressing the circuit structure. After all, if there are many-to-one or one-to-many or even many-to-many components of the same type in the path for the same starting point and ending point, the expression of the sentence content in this part will inevitably be the same, Fig. 5 reflects the phenomenon for sentence repetition.

Eventually, although one sampling can reflect the effect of the model, taking into account the influence from contingency

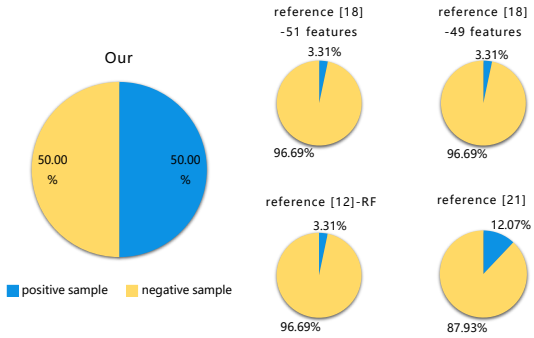

Fig. 6. The sample distribution on the comparison models.

factors, the comprehensive sampling is enough time to make the result more truly reflect the model. The paper chooses 50 random samples to form a multi-group model based on this judgment. 50 groups of sampled training data were tested on seven netlists in turn, and a total of 350 training models were generated in the experiment.

E. Evaluation indicators and experimental results

The evaluation result parameters for the experiment are derived from ML indicators, i.e., true positive rate (TPR), true negative rate (TNR), precision (PRE), accuracy (ACC), and F-measure.

Table III lists the HT detection effect of each netlist in 50 sets of training models. These experimental results are sufficient to declare that the proposed method is feasible in slightly incomplete data collection.

Study I: Sample balance and ML indicators analysis

Fig. 6 shows the division ratios for all netlist samples in five comparison models. Positive samples (PS) and negative samples (NS) respectively represent HT samples and non-HT samples. It should be noted here that the PS and NS concepts in reference [13] are contrary to all other comparison models, so the paper recalculates them. The HTtext model relies on a balanced data set for training, which makes the structure of the data set better than the other four models that depend on the wire to judge HT.

From the data comparison in Fig. 7, it is found that the proposed HTtext model obtains the best results on TPR and F-measure, and ranks 2nd on PRE. Although both TNR and ACC are the only 4th, they are only about 1 percentage point behind the leaders, which can be considered as the same level. The results of all average ML indicators are higher than 99%. The fact proves that the HTtext model is feasible. The fact proves that the HTtext model is feasible. Moreover, the overall performance for the other four models has large fluctuations compared with the paper, and their results are biased towards most non-HT samples.

F. Comparison

The obtained experimental data are compared with the four previous detection models and investigated from two aspects.

Study II: Model stabilization efficiency index

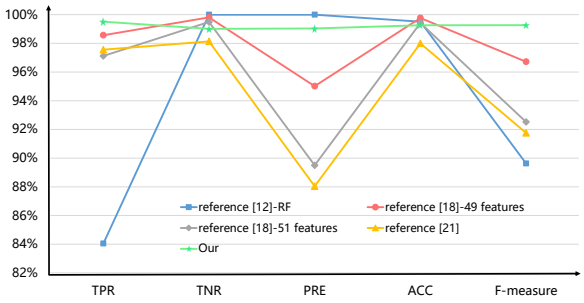

Fig. 7. Comparison results for five ML indicators.

To more effectively quantify the models high-efficiency stability in judging HT objects, this experimental group proposed the concept of stabilization efficiency index (SEI), the specific formula is as follows:

$$SE_{HT} = \frac{avg\,(TPR) + avg\,(TNR)}{\sigma\,(TPR) + \sigma\,(TNR) + 1} \tag{8}$$

where $avg\,(TPR)$ and $avg\,(TNR)$ are the mean value of TPR and TNR for each model, respectively, and $\sigma\,(TPR)$ and $\sigma\,(TNR)$ are the corresponding TPR standard deviation and TNR standard deviation.

SE_{HT} in formula (8) explains that when the ability to correctly judge PS and NS is higher and the performance fluctuation for each netlist is smaller, the model has stronger stability in generalization. The denominator is increased by 1 to avoid the amplification effect of the ratio in the case of decimals.

$$SE_{overall} = \frac{\sum_{i \in FI} avg\,(i)}{\left(\sum_{i \in FI} \sigma\,(i)\right) + 1} \tag{9}$$

Where $FI = \{TPR, TNR, PRE, ACC, F-measure\}$, $SE_{overall}$ is expanded to five ML indicators to calculate the index, according to the formula (8). This index can more comprehensively measure the generalization performance of the model. The reason for adding 1 in the denominator for formula (9) is the same as in formula (8).

In Fig. 8, the HTtext model achieved high scores of 53.27 and 71.21 on SE_{HT} and $SE_{overall}$, respectively, maintaining the best position. The other four models have dragged down their SEI evaluation due to large performance fluctuations.

Since the models for other references detect HT based on the wire features, not only rare HT samples are extracted (Fig. 6 also shows the low proportion of HT samples in these models), but their feature calculations only consider the local range for the circuit. They get fluctuating performance when using traditional ML algorithms to train on unbalanced data sets.

In the same type of word detection HT reference, Table IV shows the leading advantage of our model in all three evaluations. In addition, the comparison reference [22] only pays attention to the word expression of the local circuit, and the detection effect of the comparison model is restricted by the quality and number of HT samples.

The HTtext model's end-to-end path sentence expression depends on the word vector to include the information for

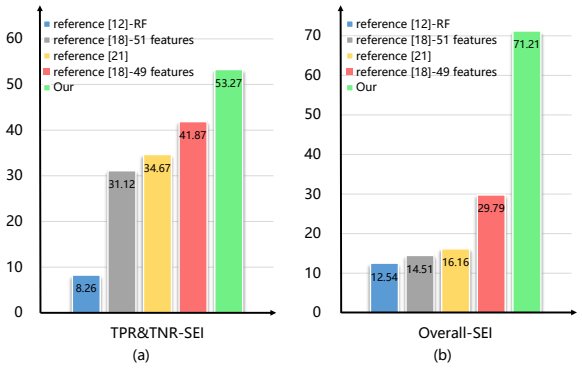

Fig. 8. Statistical values of (a) TPR&TNR-SEI (SE_{HT}) and (b) Overall-SEI ($SE_{overall}$) on each model.

the global circuit under the feature calculation of TextCNN. Also, using the simple path, it can be found that multiple path sentences are passing through the same HT component, which makes the HT information magnified to meet the balanced training for the data set. These factors are the core reasons for the stable performance of the proposed model.

G. Advantages for the HTtext model

1) the automatic single-feature: The HTtext model based on TextCNN only relies on the word vector (automatic single-feature) to solve the uncertainty problem of multi-feature.

2) HT information amplification and balanced learning: With the generating for simple path sentences, different structural information for the same HT component is obtained multiple times, which effectively amplifies the HT information. This enables the model to achieve balanced data set learning. Under equal PS and NS training, the HTtext model maintains an excellent level of high SEI.

3) golden-free chip detection: The comparison object in the detection process only relies on the Trojan-free sentence (equivalent to the golden reference) to judge the HT, which saves the hardware overhead for the "golden chip" and makes this method more cost-effective use.

VII. Conclusions

The paper successfully proves that the path information of the sentences recording circuit can detect HT effectively. These sentences are constructed by the actual signal transmission process in the port of the circuit. Learning to an automatic single feature completes the excellent detection result.

Not only does the average ACC reach 99.26%, but all the parameters are above 95%. Furthermore, the HTtext model has scores of 53.27 and 71.21 in SE_{HT} and $SE_{overall}$. The comparison shows that the proposed detection method maintains a high-balance detection level for all netlists, and the automatic single-feature detection mechanism provides the possibility for automatic feature learning.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No.61872091, No.U18042631), the Natural Science Foundation of Fujian Province, China (No.2018J01793, No.2018J01800, No.2020J01500, No.2018J01795), the Foundation of the Education Department of Fujian Province, China (No.JAT190025).

TABLE IV
Word type detecting HT between references comparison

Indicators	*Reference [22]*	*Our*
Average TPR	95.83%	**99.51%**
Average TNR	96.06%	**99.01%**
SE_{HT}	36.35	**53.27**

References

[1] M. Qiu, K. Gai, and Z. Xiong, "Privacy-preserving wireless communications using bipartite matching in social big data," *Future Generation Computer Systems*, vol. 87, pp. 772-781, 2018.
[2] A. Varas, R. Varadarajan, J. Goodrich, F. Yinug. (2021, Apr.) The Strengthening the Global Semiconductor Supply Chain in an Uncertain Era report. [Online]. Available: https://www.semiconductors.org/wp-content/uploads/2021/03/Strengthening-the-Global-Semiconductor-Supply-Chain_April-2021.pdf
[3] KKR. Choo, K. Gai, L. Chiaraviglio and Q. Yang, "A multidisciplinary approach to Internet of Things (IoT) cybersecurity and risk management," *Computers & Security*, vol. 102, pp. 1-3, 2020.
[4] 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, no. 99, pp. 10796-10826, 2020.
[5] Z. Zhang, J. Wu, J. Deng, and M. Qiu, "Jamming ACK Attack to Wireless Networks and a Mitigation Approach," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, New Orleans, LA, USA, Nov. 2008, pp. 1-5.
[6] R. Xu, L. Zhu, A. Wang, X. Du, KKR. Choo, G. Zhang, and K. Gai, "Side-Channel Attack on a Protected RFID Card," *IEEE Access*, vol. 6, pp. 58395-58404, 2018.
[7] C. Bao, D. Forte and A. Srivastava, "On application of one-class SVM to reverse engineering-based hardware Trojan detection," in *Proceedings of the 15th International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, Mar. 2014, pp. 47-54.
[8] Y. Huang, S. Bhunia, and P. Mishra, "Scalable Test Generation for Trojan Detection Using Side Channel Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 1-15, 2018.
[9] N. Shang, A. Wang, Y. Ding, K. Gai, and G. Zhang, "A machine learning based golden-free detection method for command-activated hardware Trojan," *Information Sciences*, vol. 540, pp. 292-307, 2020.
[10] C. Dong, Y. Xu, X. Liu, F. Zhang, G. He, and Y. Chen, "Hardware Trojans in Chips: A Survey for Detection and Prevention," *Sensors*, vol. 20, no. 18, pp. 5165, 2020.
[11] C. Dong, F. Zhang, X. Liu, X. Huang, W. Guo, and Y. Yang, "A Locating Method for Multi-Purposes HTs Based on the Boundary Network," *IEEE Access*, vol. 7, pp. 110936-110950, 2019.
[12] T. Kurihara, K. Hasegawa and N. Togawa, "Evaluation on Hardware-Trojan Detection at Gate-Level IP Cores Utilizing Machine Learning Methods," in *Proceedings of the 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Napoli, Italy, Jul. 2020, pp. 1-4.
[13] H. Qiu, M. Qiu, and LU. Zhihui, "Selective encryption on ECG data in body sensor network based on supervised machine learning," *Information Fusion*, vol. 55, pp. 59-67, 2020.
[14] M. Gheisari, G. Wang and MZA. Bhuiyan, "A Survey on Deep Learning in Big Data," in *IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Guangzhou, China, Aug. 2017, pp. 173-180.
[15] X. Hao, G. Zhang, S. Ma, "Deep Learning," *International Journal of Semantic Computing*, vol. 10, no. 3, pp. 417-439, 2016.
[16] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1746-1751.
[17] X. Rong, "word2vec Parameter Learning Explained," *Computer Science*, vol. 26, no. 1, pp. 48-61, 2014.
[18] C. Dong, J. Chen, W. Guo, and J. Zou, "A machine-learning-based hardware-Trojan detection approach for chips in the Internet of Things," *International Journal of Distributed Sensor Networks*, vol. 15, no. 12, pp. 1-15, 2019.
[19] M. Tomas, C. Kai, C. Greg, and D. Jeffrey, "Efficient Estimation of Word Representations in Vector Space," *Computer Science*, pp. 1-12, 2013.
[20] Trust-HUB. [Online]. Available: https://www.trust-hub.org/#/benchmarks/chip-level-trojan
[21] C. Dong, G. He, X. Liu, Y. Yang, and W. Guo, "A Multi-Layer Hardware Trojan Protection Framework for IoT Chips," *IEEE Access*, vol. 7, pp. 23628-23639, 2019.
[22] R. Lu, H. Shen, Y. Su, H. Li, and X. Li, "GramsDet: Hardware Trojan Detection Based on Recurrent Neural Network," in *2019 IEEE 28th Asian Test Symposium (ATS)*, Kolkata, India, Dec. 2019, pp. 111-116.