RLocHT: A Hardware Trojans Localization Method Utilizing Deep Learning at the Gate-Level

Yuanyuan Zhang[1], Chen Dong[1,3,*], Yi Xu[1,2], Yinan Yao[1], Chenxi Lyu[1]
1. College of Computer and Data Science, Fuzhou University, Fuzhou, China
2. Key Lab of Information Security of Network Systems (Fuzhou University), Fujian Province
3. Fujian Key Laboratory of Network Computing and Intelligent Information Processing (Fuzhou University)
nauyzhang@163.com, dongchen@fzu.edu.cn, xuyilaser@foxmail.com, 211027013@fzu.edu.cn, lvchenxi1027@foxmail.com

2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) | 978-1-6654-2207-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITAIC54216.2022.9836937

***Abstract*—Hardware is the foundation of the Cyber-Physical Systems, while the security threats brought by the hardware Trojan can make disastrous consequences. There are various studies on detecting hardware Trojans, but in order to defend against the hardware Trojans, the researches just staying at the shallow detection stage are far away from the goal. Finding the specific location of hardware Trojans is a prerequisite for more precise control of them. In this paper, a hardware Trojans localization method utilizing deep learning at the gate-level (RLocHT) is proposed. RLocHT is based on the hardware Trojans detection results of TextCNN, combined with the path division technology, which lets the convolutional neural network detect the divided paths in more detail to narrow the range of locating hardware Trojans and achieve the purpose of localization. In addition, two new evaluation metrics, localization loss rate ($LoLR$), and total locating deviation ($SD_{\triangle}$), are defined to evaluate the integrity and accuracy of RLocHT. Finally, the length of the division $cutlen$ in the path division technology is also adjusted to optimize the performance. With the optimal parameter of each netlist, for all seven benchmarks, the results achieve an average localization loss rate $LoLR_{avg}$ of 33.28% and an average total locating deviation $SD_{\triangle avg}$ of 2.55.**

***Index Terms*—hardware security, integrated circuit, hardware Trojan, deep learning, detection, localization, path division**

I. Introduction

The Cyber-Physical Systems (CPS) comprises various hardware, and integrated circuits (ICs) are the core components. Since the manufacturing process of ICs is very complex [1], producers involve third parties in the process to reduce costs [2], which poses a severe security threat [3], [4], [5], [6]. Hardware Trojan (HT) is a small piece of the circuit to achieve a malicious purpose, such as changing function, information leakage, and denial of service [7], can be inserted during its manufacturing [5], which is the primary security issue. Therefore, to guarantee the security of ICs, deep research on methods to deal with HTs is very urgent.

The production line of the IC consists of three stages: design, manufacturing, and packaging. So far, there have been many studies on detecting HTs, which can be divided into pre-silicon and post-silicon [8], [9]. Among them, the design stage is the most vulnerable to malicious attacks. If can deal with HTs between the design and manufacturing stages, a win-win situation between security and profit will be achieved. The gate-level is the last link in the design stage, and it is very effective in beating HTs at the gate-level.

Up to now, many existing works proposed methods to detect HTs. Gate-level detection is commonly used in static detection methods. Chen et al. [10] used the scoring mechanism and outlier calculation to calculate outlier values of each gate to detect HTs with netlist. Mondal et al. [11] proposed a method based on unsupervised clustering to identify networks that do not contain HT at the netlist. Dong et al. [12] extracted the netlist features using the optimization decision tree algorithm, established the HTs recognition mechanism at the gate-level and register level. Xie et al. [13] carried out a static analysis at the netlist and used Support Vector Machine to find the HTs.

Various studies on detecting HTs have been proposed, but in order to defend against HTs, just hesitating to move forward at the detection stage is not enough. Finding the specific location of HTs is a prerequisite for more precise control of them. Still, relative work has been less studied and mainly achieved indirectly with the goal of the fuzzy location while using images of ICs to detect HTs. Zhong et al. [14] used a temperature difference matrix to show the temperature difference between the chip tested and the gold chip, thus locating the HTs at a macroscopic level. Tang et al. [15] used a thermal map capture system to obtain the redundant thermal map of the chip and locate the HTs by comparing with the golden chip. Obviously, these localization results were pretty ambiguous, and could not be used to eradicate HTs.

To address the aim of locating HTs, a combination of the gate-level netlist and machine learning (ML), this paper presents for the first time a golden-free TextCNN-based method for HTs localization. The contributions are as follows.

- A path segmentation technique is proposed to divide long paths in ICs into multiple short paths, thus narrowing the range of locating HTs.
- Employ convolutional neural networks (CNN) for HTs detection, also, the CNN is used to learn the HT features and perform secondary detection on the divided short paths to achieve the goal of localization.
- Two new evaluation metrics, namely the loss of localization rate ($LoLR$) and the total locating deviation ($SD_{\triangle}$), are proposed to evaluate the effectiveness of RLocHT more comprehensively.

The remaining sections of this paper are designed as follows. The problem formulation of RLocHT is introduced in

Section II. The RLocHT is described in Section III. Section IV presents the results and analysis of the experiments, and the conclusion in Section V.

II. Problem Formulation

A. Problem description

The set of netlists $N = \{n_1, n_2, \ldots, n_i, \ldots n_l\}$, where n_i is a form of describing the layout of an IC. For n_i, the IC is composed of many circuit paths $CL^i = \{CL_1^i, CL_2^i, \ldots, CL_t^i\}$. During the manufacturing process of IC, HTs $H^i = \{h_1^i, h_2^i, \ldots, h_k^i\}$ are intentionally or unintentionally left in it. Thus, the locations of these HTs can be represented by true HT coordinates $T^i = \{(T^1_{h_1^i}, T^2_{h_1^i}), (T^1_{h_2^i}, T^2_{h_2^i}), \ldots, (T^1_{h_k^i}, T^2_{h_k^i})\}$. Moreover, $LoLR$ presents the loss of localization rate, and $SD_\triangle$ is the total locating deviation.

Where l and t respectively mean the number of netlists and circuit paths, and k is the number of HTs. The HT h_i^i at the $T^2_{h_i^i}$ th position of the $T^1_{h_i^i}$ th path is denoted as $(T^1_{h_i^i}, T^2_{h_i^i})$.

So the localization problem can be defined as

- **Input:** **(1)** Netlists N input in order. **(2)** The algorithm can generate CL^i using the netlist n_i: $CL^i = g(n_i)$, where $g(n_i)$ is the algorithm for generating the path to n_i. **(3)** The CL^i and H^i will be input together to find the T^i: $T^i = f(H^i, g(n_i))$.
- **Output:** **(1)** The result of localization P_i': $P_i' = f(H^i, g(n_i)) + e$, where e is the error. **(2)** The quantity relationship between P_i' and T^i is: $|P_i'| = (1 - LoLR) \cdot |T^i|$.
- **Objective:** Minimize $LoLR$ and $SD_\triangle$.

B. The proposed method

To solve the problem in II-A, the RLocHT model is proposed in this paper. It is the first work to achieve HTs localization at the gate-level. RLocHT combines TextCNN and path segmentation to realize the HTs pre-detection and localization. Fig. 1 shows the general flow.

Phase 1 for pre-processing , converting netlists into paths, and HTs pre-detection. Phase 2 completes the works related to HTs localization, including path division and localization. Details in the following.

III. The Specific Process of RLocHT

In this section, several critical processes of RLocHT are described, including path generation, construction of TextCNN model and HTs pre-detection, path division, and localization.

A. Path generation

This paper uses the depth-first-search method to transform the netlists into paths before starting the experiment.

The netlist describes the connection between the gates and the wires. Using wires as clues, combined with the depth-first search, can get the circuit paths. Fig. 2 is an example of path generation, (a) is a gate-level netlist, (b) is a tree diagram obtained according to (a).

A circuit path is an end-to-end sequence. Each path starts at an input port, passes through multiple gates, and ends at an output port. Since there are many input ports, output ports, and

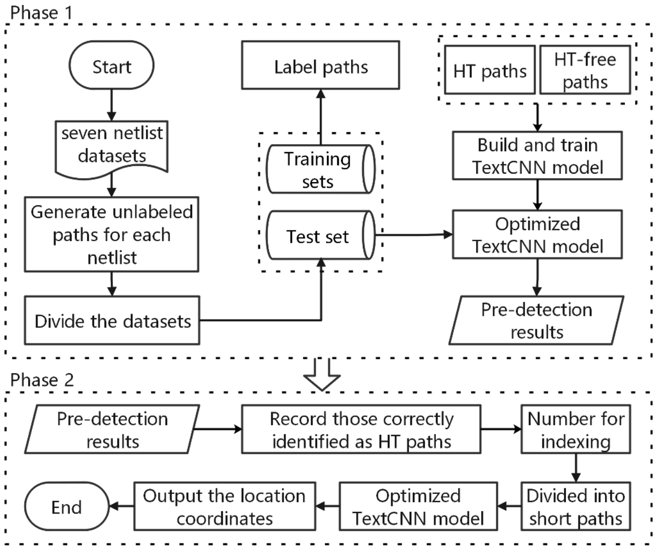


Fig. 1. The overall flow of the RLocHT.

gates, the number of paths generated for a netlist is also vast. For example, when generating paths for the netlist in Fig. 2(a), the first step is to generate its corresponding tree diagram. The root node of a tree must be a gate directly connected to an input port. In Fig. 2(b), g1 is directly connected to input ports a and b, so g1 is used as the root node, and its subsequent gates are then searched. Finally, seven paths can be obtained in Fig. 2(b), such as ['g1', 'g4', 'g7'], and the path consisting of the type names ['XOR', 'NOR', 'NAND'] can be obtained by finding their corresponding types based on the identifiers of these gates. In this example, the number of such paths consisting of type names is also seven. These two different forms of paths correspond to each other and complete the implementation of RLocHT.

Labeling the paths while generating them facilitates feature learning by TextCNN and makes predictions on unknown netlists. This paper uses seven public netlists on TrustHub.

B. Construction of TextCNN model and HTs pre-detection

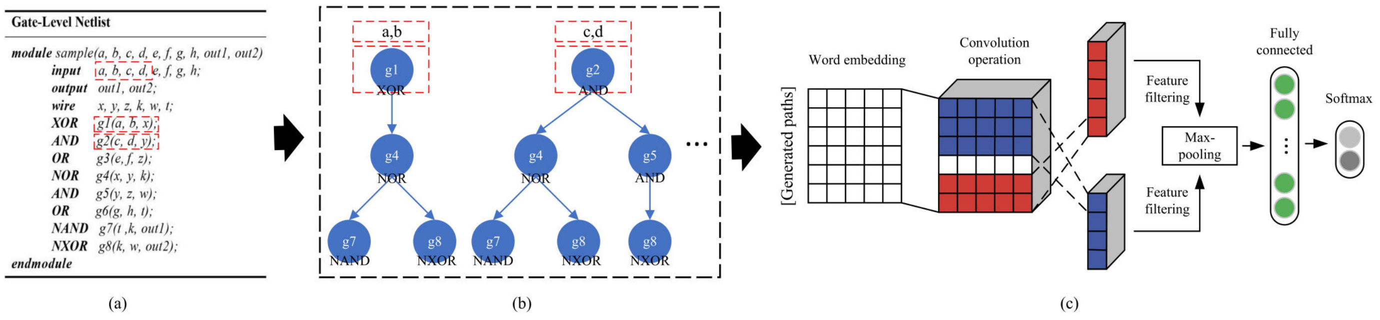
1) Model Construction: In RLocHT, the TextCNN is constructed for HTs pre-detection and localization. The structure of the TextCNN model is shown in Fig. 2(c).

Embedding layer: The embedding layer receives the input word embedding matrix. The length of matrix is equal to sentence length (sentence_len), and the width is the dimension of the word vector. Also use the padding operation to unify the size of sentences (i.e. paths) and word vectors.

Convolution layer and pooling layer: These two layer structures work together to complete the feature extraction and filtering, and finally get the most important features.

Fully connected layer: This paper uses the ReLU as the activation function of the fully connected layer and adds the dropout to avoid the overfitting.

Softmax layer: Outputs the results of the HTs pre-detection.


Gate-Level Netlist

```
module sample(a, b, c, d, e, f, g, h, out1, out2)
        input   a, b, c, d, e, f, g, h;
        output  out1, out2;
        wire    x, y, z, k, w, t;
        XOR     g1(a, b, x);
        AND     g2(c, d, y);
        OR      g3(e, f, z);
        NOR     g4(x, y, k);
        AND     g5(y, z, w);
        OR      g6(g, h, t);
        NAND    g7(t ,k, out1);
        NXOR    g8(k, w, out2);
endmodule
```

Fig. 2. An example of path generation. (a) The gate-level netlist. (b) The tree diagram. (c) The structure of the TextCNN model.

To achieve HTs localization, the storage operation is added in the last layer. It records the paths that are correctly identified as HT-inserted paths. Γ is the number of these recorded paths.

2) Model training and HTs pre-detection: Based on the work in III-A, all paths of the seven netlists are obtained. In each round of experiments, six netlists are used as the training set and the remaining one as the test set. Thus, the training set paths and labels are input together when the model is trained. Hence, the TextCNN model can learn features with HT-inserted and HT-free paths, respectively. Thus, the unknown paths can be classified.

In this method, HTs pre-detection is the basis of localization. In HTs pre-detection, the paths of the test set are input into the model, but the labels of these paths are not input, and such paths are called unknown paths. The trained model can use the learned features to classify the unknown paths. Finally, the results of the HTs pre-detection are recorded.

C. Path division

The paths pre-detected as HT-inserted are recorded to the long paths set $LL = \{LL_i | i = 1, ..., \Gamma\}$ for retrieving the HTs. In the phase 2, where the critical step is path division. RLocHT cuts the long paths to get more short paths, thus narrowing the range of locating HTs.

Algorithm 1 describes the process of path division. It takes the long paths set LL as the input, the divided short paths set SL as the output, and sets the length of the division $cutlen$. The SL_j^i means that the jth short path is divided from the LL_i, and the relationship of j and i is many-to-one. The total number of short paths $\sum_{i=1}^{TP} num_i$ is calculated by (1).

$$\sum_{i=1}^{TP} num_i = \lceil length_i / cutlen \rceil \tag{1}$$

where num_i means the number of short paths obtained after LL_i is divided, and $length_i$ is the length of LL_i.

Meanwhile, for the subsequent localization, it is necessary to set a virtual HT coordinate (V_{ij}^1, V_{ij}^2) for each short path SL_j^i to record the possible HT coordinate, where $i = 1, ..., TP; j = 1, ..., num_i$. The first and second dimension of the virtual HT coordinate can be calculated by (2),

$$\begin{cases} V_{ij}^1 = i \\ V_{ij}^2 = \begin{cases} cutlen * (t_i - \frac{1}{2}), t_i = 1, ..., num_i - 1 \\ 1 + cutlen * (t_i - 1), t_i = num_i \end{cases} \end{cases} \tag{2}$$

where t_i is the tth division of the long path LL_i.

Algorithm 1 Path division

```
Require: The long paths set $LL$, the TextCNN model
Ensure: The short paths set $SL$
 1: Initialize the length of the division $cutlen = 5$;
 2: for each $LL_i \in LL$ do
 3:     Calculate $length_i$ of $LL_i$;
 4:     $num_i = \lceil length_i / cutlen \rceil$
 5:     for $t_i = 1, j = 1; t_i \leq num_i; t_i++, j++$ do
 6:         if $t_i$ != $num_i$ then
 7:             Divide $cutlen$ gates from $LL_i$ as $SL_j^i \in SL$;
 8:             $V_{ij}^1 = i$
 9:             $V_{ij}^2 = cutlen * (t_i - \frac{1}{2})$
10:         else
11:             The remaining gates are formed into $SL_j^i \in SL$;
12:             $V_{ij}^1 = i$
13:             $V_{ij}^2 = 1 + cutlen * (t_i - 1)$
14:         end if
15:     end for
16: end for
```

D. HTs localization

Algorithm 2 describes the process of HTs localization. It takes the short paths set SL as the input, and the localization coordinates set P' as the output. In RLocHT, the principle of localization using TextCNN is to use this model to detect these divided short paths. If a short path SL_j^i is detected as HT-inserted, its (V_{ij}^1, V_{ij}^2) will become (P_{ij}^1, P_{ij}^2) and recorded into P' and finally output.

The second dimension V_{ij}^2 in (2) is explained. Since the HTs rarely appear in the input and output ports of the paths, RLocHT sets the middle location of the selected short path as the result of localization. Still, considering the possibility that the last short path obtains only one gate (which is inevitably selected to the output port at this time), when traversing the last division of a long path, the first location of the short path is selected as the locating result.

Algorithm 2 HTs localization

```
Require: The short paths set $SL$, the TextCNN model
Ensure: The localization coordinates set $P' = \emptyset$
 1: Input the short paths set $SL$ into the TextCNN model;
 2: for each $SL^i_j \in SL$ do
 3:     Vectorized $SL^i_j$ generates the embedding matrix;
 4:     Input the embedding matrix into the TextCNN model;
 5:     The contextual features $context(x)$ of $SL^i_j$ are learned;
 6:     Classify it using TextCNN;
 7:     if $SL^i_j$ is predicted as HT path then
 8:         $(V^1_{ij}, V^2_{ij}) \rightarrow (P^1_{ij}, P^2_{ij})$;
 9:         $P' = P' \cup (P^1_{ij}, P^2_{ij})$;
10:     end if
11: end for
```

IV. Experimental results and analysis

This section describes the experimental setup and the evaluation metrics and experimental results and analysis.

A. Experimental setup

1) Equipment Settings: RLocHT is implemented using python language. And on a computer system with an i5 CPU processor and 16GB of RAM.

2) Data set Settings: This paper uses seven benchmarks to verify RLocHT, and generates all their paths. The number of HT-inserted and HT-free paths is shown in Table I.

For HTs pre-detection, one benchmark is used as the test set and the remaining six as the training set. All the paths they contain are input. And for HTs localization, 7% of the Γ paths are randomly selected. There are seven rounds of experiments, and each round selects a different benchmark as the test set.

3) Experimental parameter settings: In the TextCNN model design, three convolutional kernels with different sizes 3, 4, 5, and their number of channels are set to 100. In addition, several parameters are set as follows: *dropout* = 0.5, learning rate lr = 0.001, and *num_epochs* = 5.

B. Evaluation Metrics

The evaluation metrics for HTs pre-detection are the ML indicators, while this paper focuses on the true positive rate (TPR) and the true negative rate (TNR).

Since most of the existing methods for locating HTs are implemented using circuit images, their evaluation metrics do not apply to the RLocHT. Therefore, for HTs localization, two evaluation metrics $LoLR$ and $SD_\triangle$ are designed to evaluate the effectiveness of RLocHT.

For the integrity of the localization: The localization rate LoR expresses the percentage between the total number of locating coordinates $TLC = \sum_{ij}(P^1_{ij}, P^2_{ij})$ and true HT coordinates $sum\_T = \sum_{ij}(T^1_{ij}, T^2_{ij})$. The localization loss rate $LoLR$ is calculated using (3).

$$LoLR = 1 - LoR = 1 - \frac{TLC}{sum\_T} \tag{3}$$

For the accuracy of the localization: The offset distance TOD measures the deviation between the locating coordinates and the true HT coordinates. Suppose that for a short path $\exists a, \exists b, SL^a_b \subseteq LL_a$ and its virtual HT coordinate is set to (V^1_{ab}, V^2_{ab}). When SL^a_b is determined to be HT-inserted, then $(V^1_{ab}, V^2_{ab}) \rightarrow (P^1_{ab}, P^2_{ab}) \in P'$ as a locating coordinate. The offset distance of (P^1_{ab}, P^2_{ab}) is calculated by (4), where (T^1_{ab1}, T^1_{ab}) is a true HT coordinate on SL^a_b.

TABLE I
The Benchmark Used in This Article

Benchmark	Number of HT-free Paths	Number of HT-inserted Paths
RS232-T1000	4,532,768	969,075
RS232-T1100	5,032,875	468,775
RS232-T1200	5,001,324	500,334
RS232-T1300	5,001,210	1,000,524
RS232-T1400	5,001,324	500,326
RS232-T1500	4,532,742	969,101
RS232-T1600	5,001,210	1,000,524

$$TOD_a = \frac{\sqrt{(T^1_{ab1} - P^1_{ab})^2 + (T^2_{ab1} - P^2_{ab})^2}}{cutlen} \tag{4}$$

Usually, HT is a small section of the circuit comprising multiple gates. Therefore, if (P^1_{ab}, P^2_{ab}) and multiple true HT coordinates $\{T^1_{ab}, T^2_{ab}\}$ are located in SL^a_b, it is necessary to calculate its TOD from these true HT coordinates one by one. Then the average offset distance $\overline{TOD_a}$ of (P^1_{ab}, P^2_{ab}) is calculated as shown in (5).

$$\overline{TOD_a} = avg(\sum_{z=1}^{x_{ab}} \frac{\sqrt{(T^1_{abz} - P^1_{ab})^2 + (T^2_{abz} - P^2_{ab})^2}}{cutlen}) \tag{5}$$

where x_{ab} is the number of true HT coordinates on SL^a_b.

From seven rounds of experiments, the total locating deviation $SD_\triangle$ is obtained by (6).

$$SD_\triangle = \frac{1}{N}\sum_{ij}\left(avg(\sum_{z=1}^{x_{ij}} \frac{\sqrt{(T^1_{ijz} - P^1_{ij})^2 + (T^2_{ijz} - P^2_{ij})^2}}{cutlen})\right) \tag{6}$$

A locating coordinate becomes a valid coordinate only when there exists at least one true HT coordinate that lies on the same short path as it, and N is the number of valid coordinates.

C. Experimental Result

This paper processes seven netlists, and 50 independent replicate experiments are conducted. Table II shows the experimental results obtained with different test set. Since RLocHT is the first work to implement HTs localization at the gate level and the evaluation metrics of other methods are not applicable to it, this paper is not compared with them.

In addition, the situation when different values of $cutlen$ are taken is discussed in this paper. The effect of $cutlen$ on the experimental performance is shown more visually in the form of a graph. In this paper, several netlists are randomly selected, and their cases are shown in Fig. 3.

TABLE II
EXPERIMENTAL RESULTS OF DIFFERENT TEST SET ($cutlen$=5)

Test set netlist	Total	TPR	TNR	LoLR	SD_Δ
RS232-T1000	5501843	0.9899	0.9548	0.2766	1.4218
RS232-T1100	5501650	0.9916	0.9671	0.1945	1.5032
RS232-T1200	5501658	0.9901	0.9533	0.4731	0.7801
RS232-T1300	6001734	0.9887	0.9675	0.4460	0.8669
RS232-T1400	5501650	0.9805	0.9478	0.4256	0.8703
RS232-T1500	5501843	0.9846	0.9527	0.2341	1.4068
RS232-T1600	6001734	0.9931	0.9511	0.4451	0.8829

As shown in Fig. 3, with the increase of $cutlen$, the $LoLR$ and SD_Δ decrease, and the experimental performance improve. When $cutlen$ is increased to a specific value V_{op}, the $LoLR$ and SD_Δ are minimized, and the experimental performance is optimal. So, $cutlen$ will influence the performance of RLocHT and different netlists have different V_{op}.

Therefore, the experimental results of seven netlists under their respective V_{op} are averaged, let the average localization loss rate $LoLR_{avg} = \frac{1}{7}\sum_i LoLR_i^{V_{op}}$, and the average total locating deviation $SD_{\Delta avg} = \frac{1}{7}\sum_i SD_{\Delta i}^{V_{op}}$. Ultimately, when the seven netlists are in their respective optimal $cutlen$, $LoLR_{avg}$= 33.28% and $SD_{\Delta avg}$= 2.55 can be calculated.

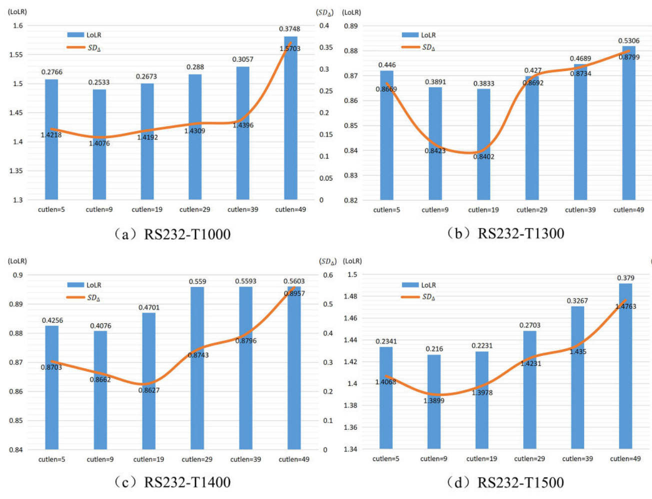


Fig. 3. The experimental performance with different cutlen.

The explanation is that the text features learned by TextCNN are the contextual features of a word. When the value of $cutlen$ is too small, the TextCNN cannot classify such short sentence well, so the experimental performance is reduced. When the value of $cutlen$ is too large, the deviation between the locating coordinate and the true HT coordinate becomes large, and the performance will also be reduced.

Since RLocHT is highly dependent on detection effectiveness, it is necessary to sacrifice some time cost to train an excellent classifier to obtain a good localization result. As for the netlist, the complete end-to-end path can include large necessary features. That is true for both small-scale and large-scale circuits. In the case of large-scale circuits, how to optimize the algorithm is the next step in this paper.

V. CONCLUSION

This paper proposes a golden-free TextCNN-based method for HTs localization at the gate-level and two new evaluation metrics. The experimental results show that the method can effectively narrow down the range of locating HTs, and the localization loss rate and the total locating deviation are within acceptable limits. Obviously, locating HTs in large netlists will be more difficult, and is also our future work.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (No.62072109, No.U1804263), Natural Science Foundation of Fujian Province (No.2021J06013, No.2018J01800, No.2020J01500, No.2018J01795).

REFERENCES

[1] 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, p. 5165, 2020.
[2] J. Chen, C. Dong, F. Zhang, and G. He, "A hardware-trojans detection approach based on extreme gradient boosting," in *2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET)*. IEEE, 2019, pp. 69–73.
[3] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, "A survey on machine learning against hardware trojan attacks: Recent advances and challenges," *Journal of IEEE Access*, vol. 8, pp. 10 796–10 826, 2020.
[4] Q. Shi, N. Vashistha, H. Lu, H. Shen, B. Tehranipoor, D. L. Woodard, and et al., "Golden gates: a new hybrid approach for rapid hardware trojan detection using testing and imaging," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2019, pp. 61–71.
[5] C. Dong, Y. Liu, J. Chen, X. Liu, W. Guo, and Y. Chen, "An unsupervised detection approach for hardware trojans," *IEEE Access*, vol. 8, pp. 158 169–158 183, 2020.
[6] Y. Liu, J. He, H. Ma, and Y. Zhao, "Hardware trojan detection leveraging a novel golden layout model towards practical applications," *Journal of Electronic Testing*, vol. 35, no. 11, pp. 529—-541, 2019.
[7] R. Sharma, V. S. Rathor, G. K. Sharma, and M. Pattanaik, "A new hardware trojan detection technique using deep convolutional neural network," *Journal of Integration the VLSI*, no. 79, pp. 1–11, 2021.
[8] G. He, C. Dong, X. Huang, W. Guo, X. Liu, and T.-Y. Ho, "Htcatcher: Finite state machine and feature verifcation for large-scale neuromorphic computing systems," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 415–420.
[9] 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, p. 1550147719888098, 2019.
[10] F. Chen and Q. Liu, "Single-triggered hardware trojan identification based on gate-level circuit structural characteristics," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.
[11] A. Mondal, R. K. Biswal, M. H. Mahalat, S. Roy, and B. Sen, "Hardware trojan free netlist identification: A clustering approach," *Journal of Electronic Testing*, no. 37, pp. 317–328, 2021.
[12] C. Dong, G. He, X. Liu, Y. Yang, and W. Guo, "A multi-layer hardware trojan protection framework for iot chips," *Journal of IEEE Access*, vol. 7, pp. 23 628–23 639, 2019.
[13] X. Xie, Y. Sun, H. Chen, and Y. Ding, "Hardware trojans classification based on controllability and observability in gate-level netlist," *Journal of IEICE Electronics Express*, vol. 14, no. 18, pp. 20 170 682–20 170 682, 2017.
[14] J. Zhong and J. Wang, "Thermal images based hardware trojan detection through differential temperature matrix," *Journal of Optik*, vol. 158, pp. 855–860, 2018.
[15] Y. Tang, L. Fang, and S. Li, "Activity factor based hardware trojan detection and localization," *Journal of Electronic Testing*, vol. 35, no. 3, pp. 293–302, 2019.