実環境を想定したトロイ回路を対象とした ランダムフォレストによるハードウェアトロイ識別

栗原 樹[1,a)]　長谷川健人[2]　福島和英[2]　清本晋作[2]　戸川 望[1]

概要：近年，IoT デバイスの普及に伴い，日常の様々なものに組み込みハードウェアが利用されている．組込み機器の需要増加により設計や製造の一部を第三者に外部委託するようになっている．これにより悪意ある第三者により回路中に悪意ある機能をもつ回路，すなわちハードウェアトロイが挿入される危険性が増大している．本稿では，ゲートレベルネットリストにおけるハードウェアトロイが持つ特徴を機械学習で学習し，ハードウェアトロイが挿入された回路のゲートレベルネットリストをトロイネット (ハードウェアトロイを構成するネット) とノーマルネット (正常な回路を構成するネット) に分類する．Trust-HUB ベンチマーク回路に加え，実環境を想定したハードウェアトロイ回路から抽出した特徴量に対し，オーバーサンプリングを適用した上でランダムフォレストで学習した．計算機実験による識別評価の結果を示す．

キーワード：ハードウェアトロイ，ゲートレベルネットリスト，機械学習，ランダムフォレスト

Hardware-Trojan Classification at Practical Trojan Netlists Utilizing Random Forests

Tatsuki Kurihara[1,a)]　Kento Hasegawa[2]　Kazuhide Fukushima[2]　Shinsaku Kiyomoto[2]
Nozomu Togawa[1]

Abstract: Recently, with the spread of Internet of Things (IoT) devices, embedded hardware devices have been used in a variety of everyday electrical items. Due to the increased demand for embedded hardware devices, some of the IC design and manufacturing steps have been outsourced to third-party vendors. Since malicious third-party vendors may insert malicious circuits, called *hardware Trojans*, into their products, developing an effective hardware-Trojan detection method is strongly required. In this paper, we learn the hardware-Trojan features in gate-level netlists with a random-forest classifier and classify the nets in the netlists into a set of normal nets and Trojan nets. We develop a machine-learning-based hardware-Trojan detection method by introducing an oversampling technique for imbalanced data and evaluate the performance of the methods using Trust-HUB benchmarks as well as several practical Trojan netlists.

Keywords: hardware Trojan, gate-level netlist, machine learning, random forest

1. はじめに

近年，Internet of Things (IoT) デバイスの普及に伴い，組込みハードウェアの需要が高まっている．組込みハードウェアをより安価で効率的に生産するため，製造拠点は国際化し，設計や製造の一部を第三者に外部委託するようになっている [1]．信頼性が十分でない第三者が設計・製造段階に関わる機会が増加したため，IC チップにハードウェアトロイ (ハードウェアに挿入された悪意のある機能を持つ回路) が挿入される危険性が指摘されている [2]–[4]．ハードウェアトロイは，システムの誤動作やパフォーマンス低下，機密情報の漏洩などを引き起こす．ハードウェアの設

[1] 早稲田大学大学院 基幹理工学研究科 情報理工・情報通信専攻
Dept. of Computer Science and Communications Engineering, Waseda University
[2] 株式会社 KDDI 総合研究所
KDDI Research, Inc.
[a)] tatsuki.kurihara@togawa.cs.waseda.ac.jp

計・製造段階には複数の工程が存在し，ハードウェアトロイはいずれの工程でも挿入される危険性がある [5]．その中でも，設計段階は Register Transfer Level (RTL) やゲートレベルネットリストといった設計データの改竄により，製造段階よりもハードウェアトロイを挿入される危険性が高い．

こうした現状を受け，ハードウェアトロイの検出に関する研究が盛んである [6]．しかし，新たなハードウェアトロイの検出方法が提案されても，すぐにそれに対抗した新たなハードウェアトロイが開発されるという問題がある [6],[7]．未知のハードウェアトロイを検出するため，機械学習を利用したハードウェアトロイの検出が不可欠となる．

本稿では，実環境を想定したハードウェアトロイ回路に対し，機械学習を用いたハードウェアトロイ検出手法を適用し，識別結果を評価する．機械学習を用いたハードウェアトロイ検出では，まず回路設計情報であるゲートレベルネットリストの各ネットに対し，ネット周辺のゲートおよびネットの情報や回路構造を特徴量として抽出する．特徴量には，対象ネット周辺のゲート種類やネットの数を参照した 11 個の特徴量 [8] および，対象ネットの構造的特徴を反映した 25 個の特徴量 [9] を使用した．計 36 個の特徴量をランダムフォレストで学習し，各ネットがハードウェアトロイを構成するネットか否かを識別する．学習の際，ノーマルネット (正常な回路を構成するネット) に対するトロイネット (ハードウェアトロイを構成するネット) の割合が平均 1.5%と低く，クラスの分布が不均衡である．不均衡データに機械学習を適用すると，少数派クラスの学習が無視されてしまうことが問題となっている [10]．そこで，特徴量データに対し，損失関数の重み付けを変化させるだけなく，オーバーサンプリングを適用することで，トロイネットの特徴量を生成し，データ数の偏りを補正した．評価する識別実験のベンチマークは，Trust-HUB [11] に公開されている 32 種類に加え，実環境での使用を想定した 22 種類を含めた 54 種類のゲートレベルネットリストを用いる．交差検証による識別実験の結果，平均 TPR (True Positive Rate) 78.1%，平均 TNR (True Negative Rate) 98.7%を達成した．

本稿の貢献を以下に示す．

(1) ランダムフォレストによるハードウェアトロイ検出手法を 54 種類のベンチマークに適用する．識別実験の結果，平均 TPR 78.1%，平均 TNR 98.7%を達成した．
(2) 特徴量データに対しオーバーサンプリングを適用することで，Trust-HUB の 32 個のネットリストについて，従来手法 [9] と比較したところ，平均 TNR を 100.0%に保ったまま，平均 TPR が 6.0 ポイント向上した．
(3) 実環境を想定したハードウェアトロイ回路に対し，使用した 20 種類全ての回路でトロイネットの検出に成功した．

2. ハードウェア設計段階における機械学習を用いたハードウェアトロイの検出

本章では，ハードウェア設計段階におけるハードウェアトロイ検出手法を紹介し，既存の手法の課題を明らかにする．近年，機械学習に基づいたハードウェアトロイ検出手法は数多く提案されている [7],[12],[13]．特に，[8],[14]–[18] では，RTL およびゲートレベルネットリストを対象としており，機械学習アルゴリズムとして，ニューラルネットワーク，サポートベクタマシン (SVM)，ランダムフォレストを用いている．

文献 [14],[15] は，ニューラルネットワークおよび SVM を用いたハードウェアトロイ検出手法を提案している．手法 [14],[15] は，ゲートレベルのネットリストに対し，各ネットがハードウェアトロイを構成するネット (トロイネット) か否か (ノーマルネット) を識別する．手法 [14] は，17 種類のネットリストに対し平均 85%のトロイネットを検出することに成功しているが，性能評価に用いられているネットリストのネット数は高々 9,000 である．

文献 [8] は，機械学習アルゴリズムにランダムフォレストを採用したハードウェアトロイ検出手法を提案している．ランダムフォレストは機械学習アルゴリズムの一つで，多数の決定木分類器の予測から多数決で予測クラスを決めるアンサンブル学習アルゴリズムである [19]．ランダムフォレストは多数の決定木分類器の予測結果を統合させることで単一の分類器より精度の高い結果を得られる．識別実験の結果，15 種類のネットリストに対し平均 TPR 68.3%，平均 TNR 100%を示している．しかし，性能評価に用いられているネットリストのネット数はこちらも高々 9,000 である．

文献 [16] は，RTL の制御フローグラフに対し，確率的ニューラルネットワークを用いてハードウェアトロイのトリガ回路を識別する．17 種類の RTL のうち，16 種類のトリガ回路を検出できており，制御フローグラフが数千のノードとエッジで構成される回路に対しては有効な手法である．

文献 [17] は，ハードウェアトロイ検出に用いるアルゴリズムとして Fine Tree, Weighted k-NN, Fine Gaussian SVM, Bagged Trees の 4 つの教師あり学習を比較している．回路の特徴量として SCOAP [20] を学習させる手法を提案しており，Bagged Trees は平均 TPR 82.5%，平均 TNR 99.0%を達成している．ベンチマークには Trust-HUB の 16 種類の回路が用いられているが，いずれも小規模である．

文献 [18] は，k-means 法によるクラスタリングと SVM を用いたハードウェアトロイ検出手法を提案している．手法 [18] は，100%の Accuracy でハードウェアトロイを検出しているが，トロイネットの特定はできていない．

表 1 ランダムフォレスト識別器の学習に用いた 36 個の特徴量.

#	特徴量	説明
1	fan_in_4	入力側 4 段手前に接続される論理ゲートの数
2	fan_in_5	入力側 5 段手前に接続される論理ゲートの数
3	in_flipflop_4	入力側 4 段手前に接続されるフリップフロップの数
4	out_flipflop_3	出力側 3 段手前に接続されるフリップフロップの数
5	out_flipflop_4	出力側 4 段手前に接続されるフリップフロップの数
6	in_loop_4	入力側で 4 段でループを構成する数
7	out_loop_5	出力側で 5 段でループを構成する数
8	in_nearest_pin	最も近いプライマリ入力の段数
9	out_nearest_pout	最も近いプライマリ出力の段数
10	out_nearest_ff	出力側で最も近いフリップフロップの段数
11	out_nearest_mux	出力側で最も近いマルチプレクサの段数
12–36	fan_in_xydy	出力側 x 段目のゲートの出力側のネットから見て入力側 y 段目のゲートに接続するネットの数 ($1 \leq x, y \leq 5$)

以上のように，文献 [8] は Accuracy 99.2%，F-measure 0.793 と最も高いスコアを出しており，ランダムフォレストはハードウェアトロイ検出において特に有効である [13]．文献 [9] は，トリガ回路の構造的特徴を反映した特徴量を提案し，32 種類のネットリストに対しランダムフォレストを用いたハードウェアトロイ検出手法を用いて平均 TPR 63.6%，平均 TNR 100%を達成している．従来手法では考慮できなかった回路の構造的特徴を学習することで，従来手法で識別できなかったハードウェアトロイ回路を識別することに成功している．いずれの手法でも性能評価には Trust-HUB のベンチマークを使用しており，より実環境に近い回路に対してもハードウェアトロイ検出手法の有効性を検討する必要がある．そこで，本稿では実環境を想定したハードウェアトロイ回路に対してランダムフォレストを用いたハードウェアトロイ検出手法を適用し，評価する．

3. ランダムフォレストを用いたハードウェアトロイ検出手法

本章では，ランダムフォレストを用いたハードウェアトロイ検出手法の詳細を示す．

3.1 検出の流れ

検出手法は，学習と識別の 2 つのフローに分かれる．学習フローでは，トロイネットの部位が予め特定されている学習データを用いて，トロイネットとノーマルネットの特徴を学習する．まず，既知のネットリストから各ネットに対してトロイネットを特徴づける 36 個の特徴量を抽出する．36 個の特徴量を表 1 に示す．特徴量には，対象ネット周辺のゲート種類やネットの数を参照した 11 個の特徴量 [8] および，対象ネットの構造的特徴を反映した 25 個の特徴量 [9] の計 36 個を使用した．次に，抽出した 36 個の特徴量をもとにランダムフォレスト識別器で学習し，パラメータを調整する．識別フローでは，学習したランダムフォレスト識別器を用いて未知のネットリストをトロイネットとノーマルネットに分類する．まず，未知のネットリストから 36 個の特徴量を抽出する．次に，学習したランダムフォレスト識別器を用いてネットリストをトロイネットかどうか識別する．

表 2 実験に使用したネットリスト.

ネットリスト	ノーマルネット	トロイネット
RS232-T1000	309	10
RS232-T1100	309	11
RS232-T1200	310	13
RS232-T1300	309	7
RS232-T1400	306	12
RS232-T1500	311	11
RS232-T1600	311	10
s15850-T100	2,420	26
s35932-T100	6,408	14
s35932-T200	6,405	12
s35932-T300	6,405	37
s38417-T100	5,799	11
s38417-T200	5,802	11
s38417-T300	5,801	44
s38584-T100	7,343	19
s38584-T200	7,373	97
s38584-T300	7,615	873
EthernetMAC10GE-T700	102,969	12
EthernetMAC10GE-T710	102,969	12
EthernetMAC10GE-T720	102,969	12
EthernetMAC10GE-T730	102,969	12
B19-T100	70,649	96
B19-T200	70,649	96
wb_conmax-T100	22,186	11
hwt1_gu	1037	1464
hwt1_ht	22	1461
hwt2_gu	1043	496
hwt2_ht	23	504
hwt3_gu	1039	424
hwt3_ht	23	428
hwt4_gu	1042	180
hwt4_ht	2	186
hwt5_gu	1040	193
hwt5_ht	2	204
hwt6_gu	1037	121
hwt6_ht	2	124
hwt7_gu	1041	1660
hwt7_ht	22	1659
hwt8_gu	1041	1672
hwt8_ht	22	1677
hwt9_gu	1055	113
hwt9_ht	22	133
hwt10_gu	3735	396
hwt10_ht	15	391
normal_blink_gu	1035	0
normal_wdp_gu	3740	0

3.2 実環境に基づいたハードウェアトロイ回路

学習に使用した 54 種類のネットリストを表 2 に示す．表 2 のうち，表上部 32 種類は Trust-HUB に公開されているネットリスト，表下部 22 種類は実回路に基づいて作成されたネットリストである．22 種類のネットリストの内訳を説明する．

表 3 実環境に基づいたハードウェアトロイ．

回路名	トリガ	ペイロード
hwt1	時間経過	ダミーレジスタ使用
hwt2	ボタン入力	情報漏えい
hwt3	時間経過	情報漏えい
hwt4	他のセンサ	性能低下
hwt5	ボタン入力	性能低下
hwt6	時間経過	性能低下
hwt7	他のセンサ	ダミーレジスタ使用
hwt8	ボタン入力	ダミーレジスタ使用
hwt9	時間経過	値の改ざん
hwt10	時間経過	情報漏えい

2 種類のノーマル回路 normal_blink_gu, normal_wdp_gu と，10 種類のハードウェアトロイ hwt$n$_ht ($1 \leq n \leq 10$) が存在し，2 種類のノーマル回路に対して 10 種類のハードウェアトロイを挿入した回路が hwt$n$_gu ($1 \leq n \leq 10$) である．ベースの回路には温湿度センサや加速度センサ，心拍数センサが搭載されており，ウェアラブル端末へのハードウェアトロイの挿入を想定している．ハードウェアトロイ hwt1–10 のトリガ回路とペイロード回路の機能を表 3 に示す．

3.3 不均衡データに対する重み付けとオーバーサンプリング

表 2 に示されるようにトロイネットの数はノーマルネットの数に比べ非常に少ない．ノーマルネットに対するトロイネットの割合が平均 1.5%と低く，クラスの分布が不均衡である．不均衡データに機械学習を適用すると，少数派クラスの学習が無視されてしまうことが問題となっている [10]．

不均衡データに対する学習では，損失関数によって少数派クラスを重み付けして学習する手法と，データ数の偏りを補正して均衡にするサンプリング手法がある．

損失関数によって少数派クラスを重み付けして学習する手法は，文献 [8], [9] の機械学習によるハードウェアトロイ検出手法でも適用されたものであり，以下のように実現されるものである．各ネットの中には，トロイネットとノーマルネットそれぞれのうちで特徴量が重複するものが存在する．そこで，特徴量が重複するネットを機械学習の訓練データから削除し，それぞれのクラスで重複する特徴量が無いよう整形する．次に，損失関数に設定する重みをクラスごとに計算する．総クラス数を c，各クラスのサンプル数を N_i としたとき，クラス k に与えられる重み w_k の計算式を式 (1) に示す．

$$w_k = \frac{\sum_{i=1}^{c} N_i}{2N_k} \tag{1}$$

したがって，トロイネットの数を N_t，ノーマルネットの数を N_n としたとき，トロイネットの重みは $(N_t+N_n)/(2*N_t)$，ノーマルネットの重みは $(N_t+N_n)/(2*N_n)$ で与えられる．ランダムフォレスト識別器では，不純度の計算にクラスの重み付けを適用することで，不均衡データを考慮した学習が可能である．トロイネットに分類されるサンプル数の割合を $p(t)$，ノーマルネットに分類されるサンプル数の割合を $p(n)$，トロイネットの重みを w_t，ノーマルネットの重みを w_n とする．不純度の指標にエントロピーを用いる場合，エントロピー I は式 (2) で計算される．

$$I = -w_t p(t) \log_2 w_t p(t) - w_n p(n) \log_2 w_n p(n) \tag{2}$$

上記に加え，本稿ではさらにデータ数の偏りを補正して均衡にするサンプリング手法を適用する．サンプリング手法には，少数派クラスのデータ数を増やすオーバーサンプリングと，多数派クラスのデータ数を減らすアンダーサンプリングがある [21]．ハードウェアトロイ検出においてはトロイネットとノーマルネットのデータ数の偏りが非常に大きいため，アンダーサンプリングを適用すると学習データの数が小さくなってしまう．アンダーサンプリングでは学習が不十分になってしまうことが問題になるため，オーバーサンプリングを採用する．

オーバーサンプリングは様々な手法が提案されている [22]．本稿では，代表的なオーバーサンプリング手法の一つである SMOTE [23] とその拡張アルゴリズムである ADASYN [24], BorderlineSMOTE [25], SVMSMOTE [26] を適用し，その中からハードウェアトロイ検出に最適なオーバーサンプリング手法を検証する．検証では，各手法でパラメータの最適化を実施し，最適化したオーバーサンプリング手法を適用した識別実験の結果を比較する．

4. 識別実験

本章では，実環境を想定したハードウェアトロイを含む 54 種類のネットリストに対し，オーバーサンプリングとランダムフォレストによるハードウェアトロイ検出手法を適用した識別実験の結果を示す．

4.1 実験環境

実装には Python 3.9.2 を使用し，scikit-learn 0.24.2 [27] のライブラリをランダムフォレストの識別器に用いた．実験にはメモリ 1.0TB，CPU として Intel Xeon Platinum 8180M を搭載する計算機を使用した．使用したベンチマークは，表 2 に示す 54 種類のネットリストである．これら 54 種類のベンチマークを学習・識別に使用し，一個抜き交差検証で識別実験をする．つまり，54 種類のベンチマークのうち，1 種類を未知のネットリストとして識別テストの対象とし，残りの 53 種類をトロイネットの場所が既知のネットリストとして学習する．全ベンチマークが識別テストの対象となるよう検証を繰り返して平均のスコアを評価する．

表 4　SMOTE のパラメータ最適化実験.

k_neighbors	TPR[%]	TNR[%]	Precision[%]	F-measure	Accuracy[%]
1	72.6	99.4	88.2	0.717	98.1
2	74.0	99.4	87.5	0.729	98.4
3	74.9	98.8	87.4	0.732	98.4
4	75.8	98.6	87.4	0.738	98.5
5	76.4	98.9	89.1	0.755	98.6
6	76.4	98.6	87.4	0.743	98.6
7	76.9	98.8	88.8	0.749	98.6
8	76.4	98.6	87.3	0.753	98.6
9	76.9	98.6	88.9	0.757	98.6
10	77.1	98.6	87.4	0.749	98.6
11	77.5	98.7	89.6	0.769	98.7
12	77.7	98.6	90.2	0.777	98.7
13	76.8	98.6	87.6	0.756	98.6
14	77.1	98.6	88.7	0.758	98.7
15	76.7	98.6	87.8	0.747	98.7

表 5　ADASYN のパラメータ最適化実験.

k_neighbors	TPR[%]	TNR[%]	Precision[%]	F-measure	Accuracy[%]
1	72.7	99.9	98.9	0.768	98.0
2	73.7	99.7	87.2	0.728	98.3
3	74.6	99.7	85.6	0.732	98.3
4	76.0	99.1	85.8	0.741	98.4
5	76.1	98.9	85.9	0.741	98.4
6	77.2	98.5	85.8	0.751	98.5
7	77.1	98.7	85.6	0.749	98.4
8	77.7	98.6	85.8	0.758	98.5
9	78.0	98.6	85.4	0.757	98.6
10	78.4	98.5	86.1	0.761	98.6
11	78.2	98.6	85.7	0.759	98.6
12	78.5	98.5	85.8	0.761	98.6
13	78.9	98.5	85.8	0.766	98.6
14	78.4	98.4	85.9	0.762	98.6
15	79.4	98.5	85.8	0.767	98.7

4.2　評価指標

本節では，実験結果の評価指標を説明する．True Negative (TN) は正しくノーマルネットとして分類されたノーマルネットの数を表す．False Positive (FP) は誤ってトロイネットとして分類されたノーマルネットの数を表す．False Negative (FN) は誤ってノーマルネットとして分類されたトロイネットの数を表す．True Positive (TP) は正しくトロイネットとして分類されたトロイネットの数を表す．True Positive Rate (TPR) はトロイネットのうち，正しくトロイネットとして分類されたものの割合を示し，TP/(TP＋FN) で定義される．True Negative Rate (TNR) はノーマルネットのうち，正しくノーマルネットとして分類されたものの割合を示し，TN/(TN + FP) で定義される．Precision はトロイネットとして分類されたネットのうち，真にトロイネットであるものの割合を示し，TP/(TP + FP) で定義される．F-measure は TPR と Precision の調和平均であり，(2·TPR·Precision)/(TPR＋Precision) で定義される．F-measure が高いほど，TPR と Precision の両方が高いといえる．Accuracy は全てのネットのうち，分類の結果が正しいネットの割合を示し，(TN＋TP)/(TN＋FP＋FN＋TP) で定義される．

4.3　オーバーサンプリング手法の最適化

まず，4 つのオーバーサンプリング手法 SMOTE, ADASYN, BorderlineSMOTE, SVMSMOTE それぞれでパラメータ最適化を実施する．最適化するパラメータは，k_neighbors である．k_neighbors は，サンプルを合成する際に何番目までの最近傍データを使用するかを指定する．k_neighbors を 1 から 15 まで変化させてランダムフォレストによるハードウェアトロイ識別実験を実施し，平均 F-measure が最良となるパラメータ値を採用する．その他のパラメータに sampling_strategy があるが，これは ‘auto’ に固定し，トロイネットをノーマルネットと同じ数までオーバーサンプリングする．ランダムフォレストのハイパーパラメータは，木の最大深さを制限なし，弱学習器の数を 500，不純度指標を情報エントロピー，クラスに対する重み調整を均等，残りは scikit-learn のデフォルト値に固定した．

表 6　BorderlineSMOTE のパラメータ最適化実験.

k_neighbors	TPR[%]	TNR[%]	Precision[%]	F-measure	Accuracy[%]
1	70.9	99.8	92.1	0.705	97.9
2	72.1	99.8	85.9	0.712	98.2
3	74.0	99.8	86.0	0.723	98.3
4	75.1	99.5	86.5	0.734	98.3
5	76.2	99.1	85.9	0.740	98.3
6	76.1	99.1	85.4	0.743	98.4
7	77.0	99.1	85.6	0.750	98.5
8	77.4	99.1	86.2	0.754	98.5
9	77.7	99.0	85.7	0.753	98.5
10	78.0	99.0	85.9	0.757	98.5
11	77.8	98.8	85.9	0.756	98.5
12	78.0	99.0	86.0	0.759	98.5
13	77.2	98.9	85.7	0.751	98.5
14	78.3	98.6	85.8	0.760	98.6
15	78.5	98.5	85.8	0.762	98.5

表 7　SVMSMOTE のパラメータ最適化実験.

k_neighbors	TPR[%]	TNR[%]	Precision[%]	F-measure	Accuracy[%]
1	71.7	99.6	92.9	0.713	98.0
2	73.3	99.0	86.6	0.719	98.3
3	74.8	98.6	86.6	0.729	98.4
4	76.0	98.6	86.0	0.738	98.5
5	76.4	98.6	87.1	0.740	98.5
6	76.9	98.5	86.3	0.747	98.5
7	76.9	98.4	85.5	0.745	98.5
8	77.6	98.4	86.0	0.751	98.5
9	77.8	98.4	85.7	0.753	98.6
10	77.9	98.4	85.5	0.753	98.5
11	78.0	98.5	85.7	0.755	98.6
12	78.0	98.4	85.9	0.756	98.6
13	78.1	98.4	85.7	0.758	98.6
14	78.5	98.4	85.8	0.761	98.6
15	78.7	98.4	85.7	0.761	98.6

表 8　オーバーサンプリング手法の比較.

アルゴリズム	TPR[%]	TNR[%]	Precision[%]	F-measure	Accuracy[%]
SMOTE	77.7	98.6	90.2	0.777	98.7
ADASYN	72.7	99.9	98.9	0.768	98.0
BorderlineSMOTE	78.5	98.5	85.8	0.762	98.5
SVMSMOTE	78.7	98.4	85.7	0.761	98.6

SMOTE の識別実験の結果を表 4 に示す．TPR, TNR, Precision, F-measure, Accuracy は，表 2 の 54 種類のネットリストの交差検証による平均スコアである．k_neighbors = 12 のとき，平均 F-measure が 0.777 と最良である．ADASYN の識別実験の結果を表 5 に示す．

表 9 ランダムフォレストのハイパーパラメータ最適化実験（オーバーサンプリングには SMOTE を利用）.

max_depth	TPR[%]	TNR[%]	Precision[%]	F-measure	Accuracy[%]
5	73.2	74.0	37.1	0.410	83.4
6	75.7	72.0	38.4	0.431	85.0
7	76.0	78.0	39.4	0.444	86.8
8	77.1	81.0	41.5	0.465	88.7
9	78.0	83.9	43.8	0.489	90.4
10	78.4	84.9	46.7	0.517	91.8
11	78.7	89.0	52.3	0.576	93.6
12	79.5	92.6	64.4	0.665	95.2
13	79.4	94.5	72.0	0.715	96.3
14	78.8	95.5	80.6	0.753	97.0
15	78.6	96.6	85.7	0.778	97.5
16	78.6	97.4	89.1	0.792	97.9
17	78.7	97.9	90.8	0.790	98.3
18	78.6	98.3	93.3	0.804	98.6
19	78.2	98.6	93.2	0.797	98.7
20	78.1	98.6	94.7	0.807	98.7
21	77.9	98.7	95.1	0.806	98.8
22	78.0	98.7	94.9	0.807	98.8
23	77.9	98.7	93.5	0.797	98.8
24	77.7	98.7	94.9	0.805	98.8
25	77.9	98.7	94.8	0.797	98.8
26	77.8	98.7	94.9	0.806	98.8
27	78.1	98.7	96.2	0.810	98.8
28	78.1	98.7	95.3	0.809	98.8
29	77.9	98.7	95.3	0.808	98.8
30	77.9	98.7	95.3	0.808	98.8

k_neighbors = 1 のとき，平均 F-measure が 0.768 と最良である．BorderlineSMOTE の識別実験の結果を表 6 に示す．k_neighbors = 15 のとき，平均 F-measure が 0.762 と最良である．SVMSMOTE の識別実験の結果を表 7 に示す．k_neighbors = 15 のとき，平均 F-measure が 0.761 と最良である．

以上の結果を踏まえて，4 つのオーバーサンプリング手法の比較を表 8 に示す．SMOTE の平均 F-measure が 0.777 と最良であり，TPR と TNR の双方で総合的に良い値を示している．ランダムフォレストによるハードウェアトロイ検出手法では，k_neighbors = 12 として，SMOTE を用いてオーバーサンプリングすることとする．また，Precision に着目すると，k_neighbors = 1 として，ADASYN によるオーバーサンプリングが最も良い．次節の実験結果では，これらのオーバーサンプリング手法を用いて，ランダムフォレストによる識別結果を評価する．

4.4 実験結果

SMOTE によるオーバーサンプリングを適用した特徴量をランダムフォレストで学習した結果を表 9 に示す．ランダムフォレストを用いた識別器に対し，木の最大深さ max_depth を 5 から 30 の間で変更し，ハイパーパラメータを最適化した．その他のハイパーパラメータは，文献 [9] に基づき，弱学習器の数を 500，不純度指標を情報エントロピー，クラスに対する重み調整を均等，残りは scikit-learn のデフォルト値に固定した．max_depth が 27 のとき，F-measure が最大値を取り，TPR と Precision がバランス良く高い値となっている．max_depth が 27 のときの識別結果の詳細を表 10 に示す．平均 TPR 78.1%，平均 TNR 98.7%を達成し，実環境に基づいたハードウェアトロイ回路を評価対象に加えても，手法 [9] と比較して平均 TNR の低下を 1.3 ポイントに抑えたまま，平均 TPR を 14.5 ポイント向上させたものとなっている．

Trust-HUB のハードウェアトロイ 32 種類に対する結果（表 2 で上側の 32 個のネットリストに対する結果）を，これらのベンチマークに対して最良のスコアを出している手法 [9] と比較する．SMOTE によるオーバーサンプリングとランダムフォレストを用いたハードウェアトロイ検出手法では，32 のベンチマークに対し，平均 TPR 69.6%，平均 TNR 100.0%を達成した．これは，手法 [9] と比較して，平均 TNR を 100.0%に保ったまま平均 TPR を 6.0 ポイント向上させたものとなっている．SMOTE によるオーバーサンプリングを適用したことで，トロイネットをより効率的に学習することに成功した．

実環境に基づいたハードウェアトロイ 22 種類に対しては，平均 TPR 88.3%，平均 TNR 96.9%，平均 Precision 98.6%，平均 F-measure 0.908，平均 Accuracy 97.8%を示した．Trust-HUB のハードウェアトロイに比較しても，ハードウェアトロイ検出手法が実環境を想定したハードウェアトロイに対し有効性が高いことが示された．

表 8 の ADASYN に注目すると，k_neighbors = 1 のとき，平均 Precision は 98.9%となっており，他のアルゴリズムと比べて 8.7 ポイント以上の差がついている．Precision を最大化することは，ハードウェアトロイ検出の実用上の観点からも重要である．ハードウェアトロイ検出手法によってトロイネットと識別されたネットに対しては，Precision が高いほど識別結果の信頼性が高い．ADASYN を適用したランダムフォレストによるハードウェアトロイ検出手法のハイパーパラメータ最適化を実施した結果，max_depth が 26 のとき，平均 TPR 73.4%，平均 TNR 99.5%，平均 Precision 99.1%，平均 F-measure 0.777，平均 Accuracy 98.0%を示した．FP の値は高々 8 で，トロイフリーのネットリスト 10 種類に対しては 9 種類のネットリストで FP が 0 となった．これは，ハードウェアトロイの挿入の有無だけを調べる場合に 54 種類中 53 種類のネットリストで識別に成功していることになる．したがって，Precision の最大化を目的にする場合 ADASYN がオーバーサンプリング手法として最も良い．

5. おわりに

本稿では，実環境での使用を想定して作成されたハードウェアトロイ 22 種類を含めた 56 種類のネットリストに対し，SMOTE によるオーバーサンプリングとランダムフォレストによるハードウェアトロイ検出手法を適用した．識別実験の結果，平均 TPR 78.1%，平均 TNR 98.7%を達成

表 10 max_depth が 27 のときのランダムフォレストによる識別結果（オーバーサンプリングには SMOTE を利用）.

分類するネットリスト	TN	FP	FN	TP	TPR[%]	TNR[%]	Precision[%]	F-measure	Accuracy[%]
RS232-T1000	309	0	0	10	100.0	100.0	100.0	1.000	100.0
RS232-T1100	309	0	0	11	100.0	100.0	100.0	1.000	100.0
RS232-T1200	310	0	0	13	100.0	100.0	100.0	1.000	100.0
RS232-T1300	309	0	1	6	85.7	100.0	100.0	0.923	99.7
RS232-T1400	306	0	0	12	100.0	100.0	100.0	1.000	100.0
RS232-T1500	310	1	0	11	100.0	99.7	91.7	0.957	99.7
RS232-T1600	311	1	1	8	88.9	99.7	88.9	0.889	99.4
s15850-T100	2420	0	0	26	100.0	100.0	100.0	1.000	100.0
s35932-T100	6409	0	7	6	46.2	100.0	100.0	0.632	99.9
s35932-T200	6405	0	11	1	8.3	100.0	100.0	0.154	99.8
s35932-T300	6405	0	0	37	100.0	100.0	100.0	1.000	100.0
s38417-T100	5799	0	10	1	9.1	100.0	100.0	0.167	99.8
s38417-T200	5801	1	10	1	9.1	100.0	50.0	0.154	99.8
s38417-T300	5798	3	9	35	79.6	100.0	92.1	0.854	99.8
s38584-T100	7344	0	15	3	16.7	100.0	100.0	0.286	99.8
s38584-T200	7316	28	99	27	21.4	99.6	49.1	0.298	98.3
s38584-T300	7344	1	1099	44	3.9	100.0	97.8	0.740	87.0
EthernetMAC10GE-T700	102969	0	1	11	91.7	100.0	100.0	0.957	100.0
EthernetMAC10GE-T710	102969	0	0	12	100.0	100.0	100.0	1.000	100.0
EthernetMAC10GE-T720	102969	0	0	12	100.0	100.0	100.0	1.000	100.0
EthernetMAC10GE-T730	102968	1	0	12	100.0	100.0	92.3	0.960	100.0
B19-T100	70649	0	0	96	100.0	100.0	100.0	1.000	100.0
B19-T200	70649	0	0	96	100.0	100.0	100.0	1.000	100.0
wb_conmax-T100	22186	0	10	1	9.1	100.0	100.0	0.167	100.0
RS232-free	303	0	0	0	-	100.0	-	-	100.0
s15850-free	2419	0	0	0	-	100.0	-	-	100.0
s35932-free	6405	0	0	0	-	100.0	-	-	100.0
s38417-free	5798	0	0	0	-	100.0	-	-	100.0
s38584-free	7343	0	0	0	-	100.0	-	-	100.0
EthernetMAC10GE-free	102944	23	0	0	-	100.0	-	-	100.0
B19-free	70618	0	0	0	-	100.0	-	-	100.0
wb_conmax-free	22182	0	0	0	-	100.0	-	-	100.0
hwt1_gu	1031	6	24	1440	98.4	99.4	99.6	0.990	98.8
hwt1_ht	22	0	1	1460	99.9	100.0	100.0	1.000	99.9
hwt2_gu	1042	1	0	496	100.0	99.9	99.8	0.999	99.9
hwt2_ht	22	1	2	502	99.6	95.7	99.8	0.997	99.4
hwt3_gu	1038	1	0	424	100.0	99.9	99.8	0.999	99.9
hwt3_ht	23	0	1	427	99.8	100.0	100.0	0.999	99.8
hwt4_gu	1039	3	1	179	99.4	99.7	98.4	0.989	99.7
hwt4_ht	2	0	0	186	100.0	100.0	100.0	1.000	100.0
hwt5_gu	1035	5	121	72	37.3	99.5	93.5	0.533	89.8
hwt5_ht	2	0	0	204	100.0	100.0	100.0	1.000	100.0
hwt6_gu	1037	0	100	21	17.4	100.0	100.0	0.296	91.4
hwt6_ht	2	0	0	124	100.0	100.0	100.0	1.000	100.0
hwt7_gu	1033	8	0	1660	100.0	99.2	99.5	0.998	99.7
hwt7_ht	22	0	1	1658	99.9	100.0	100.0	1.000	99.9
hwt8_gu	1040	1	0	1672	100.0	99.9	99.9	1.000	100.0
hwt8_ht	22	0	0	1677	100.0	100.0	100.0	1.000	100.0
hwt9_gu	1048	7	4	109	96.5	99.3	94.0	0.952	99.1
hwt9_ht	22	0	9	124	93.2	100.0	100.0	0.965	94.2
hwt10_gu	3720	15	258	138	34.9	99.6	90.2	0.503	93.4
hwt10_ht	6	9	39	352	90.0	40.0	97.5	0.936	88.2
normal_blink_gu	1030	5	0	0	-	99.5	-	-	99.5
normal_wdp_gu	3719	21	0	0	-	99.4	-	-	99.4
Average	-	-	-	-	78.1	98.7	96.2	0.810	98.8

し，従来手法に比べトロイネットの検出精度を向上させたものとなっている．本稿で導入した SMOTE とランダムフォレストによるハードウェアトロイ検出手法は，実環境を想定したハードウェアトロイに対しても有効性を発揮した．オーバーサンプリングを導入したことで，ハードウェアトロイ特有の問題である不均衡データを効率的に学習することに成功した．また，代表的なオーバーサンプリング手法の中からハードウェアトロイ検出に SMOTE が良いことを示した．一方，Precision に注目した場合には，オーバーサンプリング手法として ADASYN が最も良いこ

とも示した．本稿の結果を踏まえ，機械学習を用いたハードウェアトロイ識別の精度を向上させるのが今後の課題となる．

謝辞 本報告は総務省の「設計・製造におけるチップの脆弱性検知手法の研究開発」および総務省の令和 2 年度・令和 3 年度「ハードウェアチップの脆弱性検知手法の調査・検討等の請負」の成果の一部である．

参考文献

[1] Tehranipoor, M. and Koushanfar, F.: A survey of hardware Trojan taxonomy and detection, *IEEE Design Test of Computers*, Vol. 27, No. 1, pp. 10–25 (2010).

[2] Liu, B. and Qu, G.: VLSI supply chain security risks and mitigation techniques: a survey, *Integration, the VLSI Journal*, Vol. 55, pp. 438–448 (2016).

[3] Bhunia, S., Hsiao, M. S., Banga, M. and Narasimhan, S.: Hardware Trojan attacks: threat analysis and countermeasures, *Proceedings of the IEEE*, Vol. 102, No. 8, pp. 1229–1247 (2014).

[4] Chakraborty, R. S. and Bhunia, S.: Security against hardware Trojan through a novel application of design obfuscation, *Proc. 2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pp. 113–116 (2009).

[5] Adee, S.: The hunt for the kill switch, *IEEE Spectrum*, Vol. 45, No. 5, pp. 34–39 (2008).

[6] Elnaggar, R. and Chakrabarty, K.: Machine learning for hardware security: opportunities and risks, *Journal of Electronic Testing*, Vol. 34, No. 2, pp. 183–201 (2018).

[7] Liakos, K. G., Georgakilas, G. K., Moustakidis, S., Karlsson, P. and Plessas, F. C.: Machine learning for hardware Trojan detection: a review, *2019 Panhellenic Conference on Electronics Telecommunications (PACET)*, pp. 1–6 (2019).

[8] Hasegawa, K., Yanagisawa, M. and Togawa, N.: Trojan-net feature extraction and its application to hardware-Trojan detection for gate-level netlists using random forest, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E100.A, No. 12, pp. 2857–2868 (2017).

[9] Kurihara, T. and Togawa, N.: Hardware-Trojan classification based on the structure of trigger circuits utilizing random forests, *Proc. 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–4 (2021).

[10] Chawla, N. V., Japkowicz, N. and Kotcz, A.: Editorial: special issue on learning from imbalanced data sets, *ACM SIGKDD Explorations Newsletter*, Vol. 6, No. 1, pp. 1–6 (2004).

[11] : Trust-HUB, https://www.trust-hub.org/.

[12] Huang, Z., Wang, Q., Chen, Y. and Jiang, X.: A survey on machine learning against hardware Trojan attacks: recent advances and challenges, *IEEE Access*, Vol. 8, pp. 10796–10826 (2020).

[13] Hasegawa, K., Shi, Y. and Togawa, N.: Hardware Trojan detection utilizing machine learning approaches, *Proc. 2018 17th IEEE International Conference On Trust, Security and Privacy In Computing and Communications/ 12th IEEE International Conference On Big Data Science and Engineering (TrustCom/BigDataSE)*, pp. 1891–1896 (2018).

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

[15] Hasegawa, K., Yanagisawa, M. and Togawa, N.: A hardware-Trojan classification method using machine learning at gate-level netlists based on Trojan features, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E100.A, No. 7, pp. 1427–1438 (online), available from ⟨https://doi.org/10.1587/transfun.E100.A.1427⟩ (2017).

[16] Demrozi, F., Zucchelli, R. and Pravadelli, G.: Exploiting sub-graph isomorphism and probabilistic neural networks for the detection of hardware Trojans at RTL, *Proc. IEEE International High Level Design Validation and Test Workshop (HLDVT)*, pp. 67–73 (online), available from ⟨http://ieeexplore.ieee.org/document/8167465/⟩ (2017).

[17] Kok, C. H., Ooi, C. Y., Moghbel, M., Ismail, N., Choo, H. S. and Inoue, M.: Classification of Trojan nets based on SCOAP values using supervised learning, *Proc. 2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5 (2019).

[18] Xie, X., Sun, Y., Chen, H. and Ding, Y.: Hardware Trojans classification based on controllability and observability in gate-level netlist, *IEICE Electronics Express*, Vol. 14, No. 18, pp. 1–12 (2017).

[19] Aurelien, G.: *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly Media (2018).

[20] Goldstein, L. and Thigpen, E.: SCOAP: Sandia controllability/observability analysis program, *Proc. 17th Design Automation Conference*, pp. 190–196 (1980).

[21] Japkowicz, N.: Learning from imbalanced data sets: a comparison of various strategies, *Proc. American Association for Artificial Intelligence Workshop*, Vol. 68, pp. 10–15 (2000).

[22] Fernández, A., García, S., Herrera, F. and Chawla, N. V.: SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, *Journal of Artificial Intelligence Research*, Vol. 61, No. 1, p. 863–905 (2018).

[23] Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P.: SMOTE: synthetic minority oversampling technique, *Journal of Artificial Intelligence Research*, Vol. 16, No. 1, p. 321–357 (2002).

[24] He, H., Bai, Y., Garcia, E. and Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning, *Proc. 2008 IEEE International Joint Conference on Neural Networks*, pp. 1322–1328 (2008).

[25] Han, H., Wang, W.-Y. and Mao, B.-H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, *Proc. 2005 international conference on Advances in Intelligent Computing*, pp. 878–887 (2005).

[26] Nguyen, H. M., Cooper, E. W. and Kamei, K.: Borderline over-sampling for imbalanced data classification, *International Journal of Knowledge Engineering and Soft Data Paradigms*, Vol. 3, No. 1, p. 4–21 (2011).

[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al.: Scikit-learn: machine learning in Python, *Journal of machine Learning research*, Vol. 12, pp. 2825–2830 (2011).