Vol.2017-ARC-225 No.23
Vol.2017-SLDM-179 No.23
Vol.2017-EMB-44 No.23
2017/3/9

ネットの特徴量を用いた多層ニューラルネットワークによるハードウェアトロイ識別

長谷川 健人[1,a)]　柳澤 政生[1]　戸川 望[1]

概要：近年，IC チップの設計・製造工程における外部委託の増加に伴い，悪意ある第三者によりハードウェアトロイを挿入される可能性が指摘されている．特に設計段階でのリスクが高く，設計段階でハードウェアトロイを検出することが求められる．本稿では多層ニューラルネットワークを用いて，ゲートレベルネットリスト中のそれぞれのネットに対し，ハードウェアトロイを構成するネット (トロイネット) とそうでないネット (ノーマルネット) に分類する手法を提案する．まず，11 種類の特徴量をネットリスト中の各ネットから抽出する．抽出した特徴量にもとづき多層ニューラルネットワークを用いて機械学習することで，ネットがトロイネットか否かを分類する．提案手法では最大 100%の True Positive Rate を得て，既存のニューラルネットワークを用いたハードウェアトロイ識別手法よりも高い True Negative Rate を得た．

1. はじめに

近年，IoT をキーワードに，日常の身の回りのものにも高機能なハードウェアが搭載され，それぞれがネットワークに接続されるようになってきた．ハードウェアを安価に大量に製造するため，ハードウェア製品のセキュリティが不十分なままユーザに供給されることがある．一つのハードウェア製品を製造するため，ハードウェア設計・製造工程における第三者への外部委託や工場での製造など，多くの人々・企業を経由する機会が増加している．このハードウェア設計・製造工程のサプライチェーンにおいて，ハードウェア製品に悪意のある機能を挿入されるリスクが指摘されている [9]．ハードウェアに組み込まれた悪意のある機能は総称してハードウェアトロイと呼ばれる．ハードウェアの設計・製造には複数の工程があり，いずれの段階でもハードウェアトロイ挿入のリスクが存在する [3]．特にハードウェアの設計段階では，ハードウェア設計情報だけを改変すればハードウェアトロイを挿入できるため，他の工程でのハードウェアトロイ挿入と比較して容易である．このため，設計段階はハードウェアトロイ挿入のリスクが高い．ハードウェアトロイを抑止するため，ハードウェア設計段階におけるハードウェアトロイの検出が強く求められる．本稿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識別に着目する．

[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
[a)] kento.hasegawa@togawa.cs.waseda.ac.jp

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には，大きく分けて以下に示す二通りの目標がある．

(1) **部分検出**
ハードウェアトロイを構成するネット (トロイネット) の一部を検出する．あるいは，ネットリストにハードウェアトロイが挿入されているか否かだけを判定する．ハードウェアトロイが検出された IC チップは全体を無効とする．

(2) **全体検出**
トロイネット全体を検出する．トロイネットが検出された場合，ネットリストのうちトロイネットだけを無効化する．

(2) ではトロイネット全体を検出することを目標とするため，(2) を達成できれば (1) も達成できる．本稿では (1) を目標とする．すなわち，ネットリスト中の各ネットに対しすべてのトロイネットを正しくトロイネットと識別することを目標とする．

文献 [11] は，ゲートレベルのネットリストを対象としたハードウェアトロイ検出手法で，部分検出を目標とする．文献 [11] では，ハードウェアトロイのオンラインベンチマーク Trust-HUB [15] で公開されているネットリストからトロイネットに見られる特徴を抽出し，その特徴にもとづきハードウェアトロイを検出する．この手法では，トロイネットを含むネットリストとそうでないネットリストを識別することに成功している．文献 [11] で提案されたトロイネットの特徴は，Trust-HUB で公開されているベンチマークにもとづき手動で抽出されている．ハードウェア

Vol.2017-ARC-225 No.23
Vol.2017-SLDM-179 No.23
Vol.2017-EMB-44 No.23
2017/3/9

トロイ検出手法とハードウェアトロイの進化はいたちごっこになっていることが指摘されており [14]，新しい種類のハードウェアトロイが登場するたびにハードウェアトロイの特徴を手動で抽出することは現実的ではない．また，この手法は部分検出のため，ネットリストがトロイネットを含むか否かだけを判定している．そのため，ネットリスト中のどのネットがトロイネットかを識別することはできない．

ハードウェアトロイの全体検出を目標とする手法の例として [7] が挙げられる．この手法では，あまり使われないネットに着目してトロイネットを検出する．この手法は単に設計情報だけを利用するのではなく，シミュレーションを組み合わせてトロイネットを検出する．それゆえ，テストパターンを効率的に生成する必要があり，かつシミュレーションにも時間がかかる．

本稿ではハードウェアの設計段階におけるゲートレベルのネットリストを対象に，機械学習によるハードウェアトロイ識別を提案する．提案手法では，ネットリスト中のネットが，トロイネットか通常のネット (ノーマルネット) かを識別する．まず，予めトロイネットが特定されたネットリスト中のそれぞれのネットに対し 11 種類の特徴量を得る．次に，抽出した 11 種類の特徴量と，それぞれのネットがトロイネットかノーマルネットかを示すラベルを用いて，多層ニューラルネットワークで学習する．実験ではトロイネットが特定されたネットリストのうち，11 種類の特徴量だけを与えて，トロイネットかノーマルネットかを示す識別結果を得る．

本稿は以下のように構成される．2 章で機械学習を用いたハードウェアトロイ識別手法の研究動向を紹介し，その課題を示す．3 章で多層ニューラルネットワークを用いたハードウェア識別手法を提案する．4 章でニューラルネットワークの構造を変化させながら実験した結果を示し，その性能を評価する．また，既存手法との比較を示す．最後に 5 章で本稿をまとめる．

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識別の研究動向

本章では，機械学習を用いたハードウェアトロイ識別手法の研究動向を紹介し，既存の手法の課題を明らかにする．

2.1 ニューラルネットワークを用いた機械学習によるハードウェアトロイ識別 [5]

文献 [5] では，ゲートレベルネットリストを対象としたニューラルネットワークによるハードウェアトロイ識別手法を提案している．文献 [5] では，ニューラルネットワークの構造として，入力層，1 つの中間層，出力層からなるニューラルネットワークを適用している．ハードウェアトロイ識別のために抽出したネットの特徴量は 5 種類である．

表 1　文献 [5] で利用されたネットの特徴量.

#	Feature	特徴量
1	LGFi	ネット n から 2 段離れた論理ゲートのファンインの総数.
2	FFi	ネット n から入力側で最も近いフリップフロップまでの段数.
3	FFo	ネット n から出力側で最も近いフリップフロップまでの段数.
4	PI	ネット n からプライマリ入力までの最小の段数.
5	PO	ネット n からプライマリ出力までの最小の段数.

表 2　文献 [6] で抽出されたネットの特徴量.

#	Feature	ネット n における特徴量
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	入力側 3 段手前に接続されるフリップフロップの数
5	out_flipflop_4	入力側 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_flipflop	入力/出力側で最も近いフリップフロップの段数
11	out_nearest_multiplexer	入力/出力側で最も近いマルチプレクサの段数

抽出する 5 種類のネット特徴量とその説明を表 1 に示す．入力層からは 5 種類の特徴量に対応する値を入力する．中間層は 1 層で，7 ユニットの構造を適用している．出力層からは 0 から 1 までの値が出力される．出力層の出力値によりネットがトロイネットか否かを識別する．

文献 [5] では，トロイネットを正しくトロイネットと識別したことを示す指標である True Positive Rate (TPR) において最大で 100%を得ているが，平均 TPR は 81%にとどまり，平均 TPR のさらなる向上と，ノーマルネットを正しくノーマルネットと識別する必要がある．

2.2 機械学習のためのネットリスト特徴抽出 [6]

文献 [6] では，まずネットリストからハードウェアトロイを識別するために有効と考えられる 51 種類の特徴量を抽出する．抽出した特徴量に関して，機械学習アルゴリズムの 1 つである Random Forest を用いて，機械学習によるハードウェアトロイ識別に有効な 11 種類の特徴量を抽出している．表 2 に抽出された 11 種類の特徴量を示す．

この手法では，TPR において最大で 100%の結果を得ている．さらに，ノーマルネットを正しくノーマルネットと識別した割合 (TNR) も最大で 100%を得ている．しかし，すべてのベンチマークの平均 TPR の観点では既存の機械学習を用いた手法よりも低い．

2.3 既存手法の課題と提案手法のポイント

機械学習では TPR と TNR のトレードオフが存在するため，両者を同時に最大化するようなハードウェアトロイ識別手法を提案するのは難しい．ハードウェアトロイ識別において，1 章で議論した通り，本稿ではネットリスト中のすべてのトロイネットを正しくトロイネットと識別することを第一の目標と定める．その上で，TNR をできるだけ向上させる．これを達成するため，以下の点に着目する．

Vol.2017-ARC-225 No.23
Vol.2017-SLDM-179 No.23
Vol.2017-EMB-44 No.23
2017/3/9

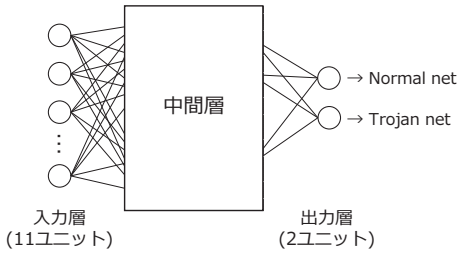

図 1 提案する多層ニューラルネットワークの構成.

(1) **11 種類の特徴量を適用**
文献 [6] において，Random Forest を用いた手法によりハードウェアトロイ識別のための 11 種類の特徴量が提案されている．これらの特徴量を用いることで，トロイネットとノーマルネットを精度よく識別できると考えられる．

(2) **ニューラルネットワークの多層化**
文献 [5] では単層のニューラルネットワークを利用している．ニューラルネットワークは単層でもモデルを十分に表現できることが知られているが，多層化することでより柔軟にモデルを表現できる．本稿ではニューラルネットワークを多層化してハードウェアトロイを識別する．

(3) **評価指標の考慮**
ニューラルネットワークを多層化する際，評価指標として TPR と TNR の両方を考慮する．

以上にもとづき，本稿では多層ニューラルネットワークを用いたゲートレベルネットリストに対するハードウェアトロイ識別手法を提案する．

3. 多層ニューラルネットワークによるハードウェアトロイ識別

3.1 ハードウェアトロイ識別に利用する特徴量

提案手法では，文献 [6] で提案された表 2 に示す 11 種類の特徴量を適用する．ゲートレベルのネットリスト中の各ネットから表 2 に示す特徴量を抽出し，多層ニューラルネットワークで学習する．

3.2 ニューラルネットワークの層の構成

中間層が複数であるニューラルネットワークを多層ニューラルネットワークとする．本節では，どのように多層ニューラルネットワークの構造を決定するかを説明する．

多層ニューラルネットワークにおいて，その構造は中間層の数，それぞれの中間層におけるユニット数など，様々な条件を検討する必要がある．ニューラルネットワークの構造は識別性能を大きく左右する．入力層と出力層はデータ構造に依存して決まるが，最適な中間層の数やユニット数を決める理論的な手法は存在しない [4]．本節では入力層と出力層，中間層の数とユニット数に分けて説明する．

3.2.1 入力層と出力層

入力層と出力層のユニット数は，分類するデータの構造に依存する．

入力層のユニット数は，ネットの特徴量の数に対応する．そのため，提案手法では入力層のユニット数を 11 とする．それぞれのユニットに対し，それぞれの特徴量を入力して学習を始める．

出力層のユニット数は，分類方法によっていくつか存在する．ネットをトロイネットとノーマルネットに分類する二値分類では，大きく二通りの構造が考えられる．一つ目は，出力層のユニット数を 1 つとし，その出力値に応じてクラスを決定する手法である．例えば出力値における閾値を 0.5 とした時，0.5 未満の出力をノーマルネット，0.5 以上の出力をトロイネットと考えることができる．二つ目は，出力層のユニット数を二つとし，一つ目のユニットをノーマルネット，2 つ目のユニットをトロイネットに対応させる手法である．二つのユニットの出力値のうち大きい値を出力した方を，分類されたクラスと見なす．一つ目の手法では，得られる出力値が一つのため，分類結果における TPR と TNR を調整するためにはノーマルネットかトロイネットかを識別するための閾値を変化させることしかできない．一方で二つ目の手法では，一般的により大きい値を出力するユニットを分類結果とするが，その出力値がどのくらいかを詳細に検討することも可能となる．そこで，提案手法では二つ目の手法を適用する．

3.2.2 中間層のユニット数と中間層の数

一般に，中間層の数と中間層のユニット数がニューラルネットワークの識別性能を左右することが知られている．また，中間層の数やユニット数が増加させるとニューラルネットワークがより柔軟に学習できることも知られている [10]．一方で，中間層の数やユニット数を増加することにより，計算時間が増大する．また，ニューラルネットワークが過学習を起こす可能性が指摘されている．さらに，中間層の数が多くなると，学習過程の一つである誤差逆伝播において，誤差が入力層に近い層にまでうまく伝わらない勾配消失問題 [1] も発生する．

本稿では 4 章で示すように以下の手順に従い実験し，中間層のユニット数と中間層の数を決定する．

(1) 中間層を 1 層とし，ユニット数を 10, 12, 15, 20, 50, 100, 200, 500 で変えながら実験.
(2) 中間層を 2 層とし，1 層目のユニット数を 20, 50, 100, 200，2 層目のユニット数を 10, 12, 15, 20, 50, 100, 200, 500 で変えながら実験.
(3) 中間層を 3 層とし，1, 3 層目のユニット数を 50, 100, 200，3 層目のユニット数を 50, 100, 200 で変えながら実験.

提案するニューラルネットワークの構造を図 1 に示す．

Vol.2017-ARC-225 No.23
Vol.2017-SLDM-179 No.23
Vol.2017-EMB-44 No.23
2017/3/9

入力層は 11 ユニット，出力層は 2 ユニットである．中間層は 4 章で検討する．

3.2.3 誤差関数を用いた重みづけ

トロイネットの数はノーマルネットの数と比較して少ない．表 3 に示されるように，例えば RS232-T1000 と呼ばれるベンチマークデータではノーマルネットの数は 283，トロイネットの数は 36 である．機械学習ではデータの比重をそろえる必要がある．文献 [5] ではトロイネットのデータを複数回学習している．例えば RS232-T1000 の場合，$283/36 = 7.86\ldots \simeq 8$ 回学習する．しかし，この手法では同じデータを複数回学習するため，より多くの時間がかかる問題がある．

提案手法では，ニューラルネットワークの誤差関数に重みづけすることでデータの比重をそろえる．ニューラルネットワークの誤差関数として，一般的に利用される交差エントロピーを提案手法でも適用する [10]．その中でも特に重み付き交差エントロピーを利用する [13]．2 次元の出力層の出力を (t_1, t_2)，そのときの正解データを (y_1, y_2)，それぞれの重みを (w_1, w_2) としたとき，重み付き交差エントロピー E は次式で示される．

$$E = -\sum_{k=1}^{2} w_k \cdot t_k \ln y_k \qquad (1)$$

提案手法では，重みの係数をトロイネットの数とノーマルネットの数の比で定める．例えば，RS232-T1000 では $(1, 7.86\cdots)$ となる．ノーマルネットに対する誤りに比べ，トロイネットで誤りが生じれば交差エントロピーはより大きく変化する．これは誤差に対するペナルティが大きいことを意味する．

3.3 多層ニューラルネットワークの構造の評価

本節では，多層ニューラルネットワークによるハードウェアトロイ識別手法を示す．また，その性能の評価方法を説明する．

実験では，表 3 に示す Trust-HUB [15] で公開されている 17 種類のデータを使用する．Trust-HUB のデータはトロイネットとノーマルネットが特定されている．これらのデータに対し，Leave-one-out 交差検証 [8] を適用する．Leave-one-out 交差検証では，17 種類のうち 1 種類を残して 16 種類を学習する．残した 1 種類のネットリストを分類し，正解と比較する．17 種類すべての平均 TPR と平均 TNR を指標とする．

機械学習の性能を評価するための指標はいくつか存在する．

真のトロイネット (Trojan net) を正しくトロイネットと識別した場合を True Positive (TP)，誤ってノーマルネット (Normal net) と識別した場合を False Negative (FN) と定義する．同様に，真のノーマルネットを正しくノーマルネットと識別した場合を True Negative (TN)，誤ってトロイネットと識別した場合を False Positive (FP) と定義する．

表 3 実験で使用した Trust-HUB ベンチマーク．

Data name	# of normal nets	# of Trojan nets
RS232-T1000	283	36
RS232-T1100	284	36
RS232-T1200	289	34
RS232-T1300	287	29
RS232-T1400	273	45
RS232-T1500	283	39
RS232-T1600	292	29
s15850-T100	2,429	27
s35932-T100	6,407	15
s35932-T200	6,405	12
s35932-T300	6,405	37
s38417-T100	5,798	12
s38417-T200	5,798	15
s38417-T300	5,801	44
s38584-T100	7,343	19
s38584-T200	7,373	97
s38584-T300	7,614	874

これら 4 つの指標に対し，次の 4 通りの指標を定義する．真のトロイネット全体 (= FN + TP) のうち正しくトロイネットと識別した数 (= TP) の割合を True Positive Rate (TPR) と定義する．TPR は再現率 (Recall, R) とも呼ばれる．同様に，真のノーマルネット全体 (= FP + TN) のうち正しくノーマルネットと識別した数 (= TN) の割合を True Negative Rate (TNR) と定義する．

本稿では，1 章で説明した通り，トロイネット全体を検出することを目的とする．このため，すべてのトロイネットを正しくトロイネットと識別する TPR を最大化することが第一目標となる．しかし，TPR だけを最大化するのは実用的ではない．例えばネットリスト中のすべてのネットをトロイネットと識別すれば，TPR は 100%となる．そのため，2 つ目の指標として，TNR を最大化する．

4. 評価実験

本章では，3 章にもとづく実験結果と既存手法との比較を示す．

実験には Trust-HUB [15] で公開されるベンチマークのうち，17 種類のゲートレベルのネットリストを利用した．実験に利用したハードウェアトロイのベンチマークを表 3 に示す．実験には CPU として Intel Xeon E5-2695 v4，メモリ 256GB を搭載したコンピュータを使用した．実装は Python3 を用いて，機械学習のライブラリとして Chainer [12] を利用した．

4.1 実験結果

3 章にもとづき，中間層のパラメータを変えながら実験した．中間層のパラメータを変化させたときの結果を示す．

実験 1：中間層が 1 層のとき

Vol.2017-ARC-225 No.23
Vol.2017-SLDM-179 No.23
Vol.2017-EMB-44 No.23
2017/3/9

表 4　実験結果 (中間層: 1 層).

# of units (1st layer)	Average values TPR	TNR
10	85.9%	59.7%
12	85.7%	58.5%
15	82.7%	59.5%
20	87.4%	59.4%
50	84.2%	62.9%
100	83.4%	60.1%
200	86.8%	62.3%
500	86.7%	53.3%
Average	85.3%	59.5%

表 5　実験結果 (中間層: 2 層).

# of units (1st layer)	(2nd layer)	Average values TPR	TNR	# of units (1st layer)	(2nd layer)	Average values TPR	TNR
20	10	87.4%	58.4%	100	10	79.1%	61.4%
	12	83.5%	61.2%		12	85.8%	57.3%
	15	84.7%	56.4%		15	82.1%	61.4%
	20	88.0%	59.6%		20	87.7%	58.6%
	50	83.7%	59.9%		50	81.9%	61.1%
	100	83.9%	57.5%		100	83.7%	59.8%
	200	82.5%	65.7%		200	79.1%	66.6%
	500	77.9%	59.9%		500	85.4%	60.6%
	Average	83.9%	59.8%		Average	83.1%	60.8%
50	10	84.1%	59.5%	200	10	85.7%	61.8%
	12	83.3%	57.2%		12	82.6%	58.6%
	15	85.0%	62.0%		15	77.1%	62.4%
	20	83.9%	59.9%		20	79.3%	66.7%
	50	86.0%	60.2%		50	84.7%	66.8%
	100	88.9%	59.2%		100	84.8%	60.2%
	200	84.2%	57.6%		200	78.2%	62.3%
	500	82.2%	58.6%		500	83.6%	66.0%
	Average	84.7%	59.3%		Average	82.0%	63.1%

中間層の数を 1 層にしたときの結果を表 4 に示す．それぞれのユニット数における TPR, TNR の値を示す．TPR, TNR それぞれにおける最大値を下線で示す．

実験結果より，ユニット数が 20 のとき TPR が最大となり，50 のときに TNR が最大となることが分かる．TPR が 86%を超えるのはユニット数が 20, 200, 500 のときで，比較的ユニット数が多いときである．一方，TNR が 60%を超えるのはユニット数が 50, 100, 200 のときである．ただし，TPR, TNR ともに最大となるのはユニット数が 500 のときでないことから，単純にユニット数を増加させれば良い訳ではないことが分かる．

実験 2：中間層が 2 層のとき

中間層の第一層のユニット数をそれぞれ 20, 50, 100, 200 に固定し，中間層の第二層のユニット数を変えながら実験した．なお，中間層の第二層のユニット数は，10, 12, 15, 20, 50, 100, 200, 500 に変えながら実験した．

中間層の数を 2 層にしたときの結果を表 5 に示す．それぞれのユニット数における TPR, TNR の値を示す．TPR, TNR における最大値を下線で示す．

実験結果より，(第一層のユニット数, 第二層のユニット数)=(50, 100) のとき TPR が最大，(200, 50) のとき TNR が最大となった．各組の平均値を見ると，中間層を 1 層だけにしたときよりも平均 TPR は減少している．しかし，第一層を 200 ユニットとした中間層 2 層の組では平均 TNR が 63.1%と，中間層 1 層だけのときよりも上回っている．

実験 3：中間層が 3 層のとき

中間層の第一層のユニット数をそれぞれ 50, 100, 200，第二層のユニット数をそれぞれ 20, 50, 100, 200 と固定し，中間層の第三層のユニット数を変えながら実験した．中間層の第三層のユニット数は，50, 100, 200 に変えながら実験した．

中間層の数を 3 層にしたときの結果を表 6 に示す．それぞれのユニット数における TPR, TNR の値を示す．TPR, TNR における最大値を下線で示す．

実験結果より，ユニット数の組み合わせが (100, 20, 50) のとき TPR が最大となり，(200, 200, 100) のとき TNR が最大となる．

ここで，TPR を最大化した場合に TNR をできるだけ大きくすることを考える．表 6 の平均値より，TNR は少なくとも 70%程度であることが期待される．TPR を上から順に考えたとき，ユニット数が (200, 100, 50) のとき (波線部)，TPR が大きく，かつ TNR が 70%を超える．したがって，ユニット数 (200, 100, 50) がハードウェアトロイ識別に最も適した構造であると言える．

4.2　既存手法との比較

既存手法と比較する．比較するのは，提案手法にもとづく実験結果のうち，TPR を最大化した上で TNR をできるだけ大きくした構造 (中間層 3 層，ユニット数 (200, 100, 50)) とする．文献 [6] は，Random Forest を用いた手法である．文献 [5] は，ニューラルネットワークを用いた手法である．文献 [2] は，回路の相互相関にもとづきクラスタリングする手法である．これらの手法と TPR, TNR を比較した結果を表 7 に示す．文献 [2] は同一のベンチマークと比較している．

TPR に着目すると，提案手法は平均 TPR は既存のいずれの手法よりも上回っている．個別のベンチマークでは，s15850-T100 に関していずれのベンチマークを上回っている．RS232 系ではすべてのベンチマークで提案手法の TPR が 90%以上となっている．文献 [2] との比較では，個別のベンチマークにおいて 4 個中 2 個のベンチマークで TPR が上回る結果を得た．

一方，TNR に着目すると，提案手法では平均 TNR として 70%を得た．既存手法のうち平均 TPR が最も高いニューラルネットワークを用いた手法 [5] の TNR は 69%であるため，これを上回る結果となった．TPR を第一に見て TNR をできるだけ高くすることを考えたとき，提案手法は最も良い結果を得たと言える．さらに，提案手法では s15850-T100–s38584-T300 において，ほとんどのベンチマークで 90%を超える TNR を得た．これらのベンチマークは RS232 系のベンチマークと比べネット数が多く，そのうちの 9 割以上のノーマルネットを正しくノーマルネット

Vol.2017-ARC-225 No.23
Vol.2017-SLDM-179 No.23
Vol.2017-EMB-44 No.23
2017/3/9

表 6　実験結果 (中間層: 3 層).

# of units			Average values		# of units			Average values		# of units			Average values	
(1st layer)	(2nd layer)	(3rd layer)	TPR	TNR	(1st layer)	(2nd layer)	(3rd layer)	TPR	TNR	(1st layer)	(2nd layer)	(3rd layer)	TPR	TNR
50	20	50	82.2%	58.4%	100	20	50	86.2%	60.4%	200	20	50	82.7%	65.3%
		100	81.9%	60.4%			100	85.2%	64.4%			100	80.7%	68.8%
		200	83.7%	59.9%			200	79.6%	63.4%			200	80.5%	67.9%
	50	50	78.2%	61.9%		50	50	83.2%	59.9%		50	50	80.6%	75.4%
		100	81.5%	55.7%			100	81.8%	63.3%			100	74.3%	61.0%
		200	82.9%	60.1%			200	78.1%	61.8%			200	83.8%	64.8%
	100	50	77.5%	67.1%		100	50	79.7%	60.0%		100	50	84.8%	70.1%
		100	80.6%	60.8%			100	77.3%	59.7%			100	76.6%	70.0%
		200	75.6%	61.4%			200	78.9%	67.2%			200	80.2%	69.4%
	200	50	79.2%	64.9%		200	50	79.1%	60.8%		200	50	77.3%	70.3%
		100	76.8%	65.7%			100	83.9%	66.7%			100	81.0%	75.7%
		200	77.8%	67.0%			200	78.4%	63.5%			200	79.9%	60.5%
	Average		79.8%	61.9%		Average		81.0%	62.6%		Average		80.2%	68.3%

表 7　既存手法との比較.

	TPR				TNR			
Data name	[6]	[5]	[2]	Ours	[6]	[5]	[2]	Ours
RS232-T1000	100%	40%	-	100%	98%	67%	-	24%
RS232-T1100	58%	100%	-	78%	99%	62%	-	25%
RS232-T1200	85%	70%	-	91%	100%	52%	-	55%
RS232-T1300	97%	22%	-	86%	99%	73%	-	65%
RS232-T1400	100%	100%	-	100%	100%	51%	-	15%
RS232-T1500	95%	67%	-	82%	99%	66%	-	47%
RS232-T1600	90%	78%	-	97%	98%	64%	-	28%
s15850-T100	33%	89%	61%	81%	100%	76%	99%	96%
s35932-T100	53%	100%	-	80%	100%	85%	-	99%
s35932-T200	8%	88%	27%	67%	100%	83%	99%	88%
s35932-T300	92%	100%	-	100%	100%	59%	-	97%
s38417-T100	42%	100%	100%	83%	100%	72%	99%	98%
s38417-T200	33%	73%	-	93%	100%	67%	-	74%
s38417-T300	98%	75%	-	100%	100%	76%	-	94%
s38584-T100	16%	100%	-	16%	100%	56%	-	99%
s38584-T200	-	93%	99%	91%	-	83%	98%	93%
s38584-T300	-	89%	-	97%	-	76%	-	93%
Average	67%	81%	(72%)	85%	99%	69%	(99%)	70%

と識別できたと言える．提案手法により，既存のニューラルネットワークを用いたハードウェアトロイ識別手法を改善できたことを確認できる．

5. おわりに

本稿では，多層ニューラルネットワークを用いたハードウェアトロイ識別手法を提案した．本稿の実験結果により，既存のニューラルネットワークを適用した手法を上回ることを確認した．今後，ニューラルネットワークを用いたハードウェアトロイ識別手法の改良に加え，ハードウェアトロイの局所性を始めとする特徴を踏まえた識別手法を検討する．

謝辞

本研究開発は一部，総務省 SCOPE (受付番号 141303001) の委託を受けた．

参考文献

[1] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[2] B. Cakir and S. Malik, “Hardware trojan detection for gate-level ICs using signal correlation based clustering,” in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 471–476, 2015.

[3] J. Francq and F. Frick, “Introduction to hardware trojan detection methods,” in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 770–775, 2015.

[4] L. Hansen and P. Salamon, “Neural network ensembles,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[5] 長谷川健人, 柳澤政生, 戸川望, “ニューラルネットを利用したネットリストの特徴にもとづくハードウェアトロイ識別,” **信学技報**, vol. 116, no. 93, CAS2016-1, pp. 1–6, 2016.

[6] ——, “Random Forest を用いたネットリスト特徴選択と機械学習によるハードウェアトロイ識別,” *DA* **シンポジウム** *2016* **論文集**, vol. 2016, no. 3, pp. 8–13, 2016.

[7] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, “Overcoming an untrusted computing base: detecting and removing malicious hardware automatically,” in *Proc. Symposium on Security and Privacy (SP)*, pp. 159–172, 2010.

[8] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Proc. International Joint Conference on Artificial Intelligence*, vol. 2, pp. 1137–1143, 1995.

[9] B. Liu and G. Qu, “VLSI supply chain security risks and mitigation techniques: A survey,” *Integration, the VLSI Journal*, pp. 1–10, 2016.

[10] 岡谷貴之, **深層学習**． 講談社, 2015.

[11] M. Oya, N. Yamashita, T. Okamura, Y. Tsunoo, M. Yanagisawa, and N. Togawa, “Hardware-Trojans rank: quantitative evaluation of security threats at gate-level netlists by pattern matching,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E99.A, no. 12, pp. 2335–2347, 2016.

[12] S. Tokui, K. Oono, S. Hido, and J. Clayton, “Chainer: a next-generation open source framework for deep learning,” in *Proc. Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

[13] T. Zahavy, A. Magnani, A. Krishnan, and S. Mannor, “Is a picture worth a thousand words? A deep multi-modal fusion architecture for product classification in e-commerce,” pp. 1–9. http://arxiv.org/abs/1611.09534 nov 2016.

[14] J. Zhang, F. Yuan, and Q. Xu, “DeTrust: defeating hardware trust verification with stealthy implicitly-triggered hardware Trojans,” in *Proc. ACM SIGSAC Conference on Computer and Communications Security (ACM-CCS)*, pp. 153–166, 2014.

[15] “Trust-HUB.” http://www.trust-hub.org