グラフ学習を用いたゲートレベルハードウェアトロイ識別に対するメンバシップ推論攻撃

山下 一樹[1,a)]　長谷川 健人[2]　披田野 清良[2]　福島 和英[2]　戸川 望[1]

概要：ハードウェアトロイとは，IC の設計・製造工程で悪意ある第三者により挿入された不正回路であり，その検知に，ゲートレベルネットリストのグラフ学習に基づく手法が有効であると報告されている．一方で，機械学習モデルに対するメンバシップ推論攻撃の危険性が指摘されている．この攻撃手法は，あるデータが識別器の訓練データとして使用されたかを推論する攻撃手法であり，攻撃が成功した場合，悪意ある第三者に訓練データが漏洩することになる．本稿ではグラフ学習を用いたハードウェアトロイ識別器を対象としたメンバシップ推論攻撃を提案する．提案手法は，まず攻撃対象となるハードウェアトロイ識別器に訓練データに含まれるハードウェアトロイを与えたときの出力値と，訓練データに含まれないハードウェアトロイを与えたときの出力値を得る．そして，これらの出力値の差異を学習することで，訓練データに含まれるハードウェアトロイの構造を明らかにする．評価実験の結果，攻撃対象のハードウェアトロイ識別器に対し，訓練データのハードウェアトロイのデータと，訓練データ以外のハードウェアトロイのデータを，AUC 0.988 の精度で識別できることを確認した．

キーワード：ハードウェアトロイ，グラフ学習，メンバーシップ推論攻撃，ネットリスト，論理ゲート

Membership Inference Attack on Hardware-Trojan Detection Utilizing Graph Neural Networks at Gate-Level Netlists

Kazuki Yamashita[1,a)]　Kento Hasegawa[2]　Seira Hidano[2]　Kazuhide Fukushima[2]
Nozomu Togawa[1]

Abstract: Recently, the threat of injecting a circuit with malicious functions called a hardware Trojan has been growing. Hardware Trojans can be effectively detected using machine learning and a method using graph neural networks (GNNs) has been proposed. On the other hand, the risk of membership inference attacks against machine learning models has been pointed out. This attack infers whether or not certain data is used as training data for a model. The success of the attack implies a privacy violation against the data providers and an attacker definitely knows which Trojans are used or not. In this paper, we propose a membership inference attack method for the hardware-Trojan classifier based on GNN and confirm the possibility that the training data can be exposed.

Keywords: hardware Trojan, graph neaural network, membership inference attack, netlist, logic gate

1. はじめに

近年，様々なモノの IoT (Internet of Things) 化が進んだことで IC の需要が高まり，ハードウェア製品の開発においては IC を安価で効率的に生産することの重要性が増している．安価で効率的に IC を生産するための方法とし

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

て，設計や製造工程の一部をサードパーティに委託する手段が用いられている [1]．ハードウェアの設計・製造段階では，設計書の作成に始まり，IP コアの設計やレイアウト設計，製造など様々な段階が存在しており，各段階でサードパーティが関与する可能性がある．

ハードウェア製造の過程にサードパーティが関与する中で，悪意のある第三者にハードウェアトロイを挿入されるリスクが報告されている [2], [3]．ハードウェアトロイとは，回路の内部状態がある条件を満たした時に，情報漏洩や機能改変などの製造元が意図しない動作を発現する回路である．動作が発現した場合，IC 製品に甚大な被害を引き起こすため，システムが製品化される前に検知する必要がある．しかし，ハードウェアトロイは一般的に小規模かつ極めてまれな条件においてのみ起動する回路であるため，検知が困難である．

本稿では，回路の設計段階における識別手法に焦点を当てる．回路の設計段階は簡単にハードウェアトロイを挿入できるために，攻撃者に狙われやすい [2]．また，回路の設計段階でハードウェアトロイを識別できれば製造段階や製造後にハードウェアトロイを除去する必要がなくなるため，設計段階でハードウェアトロイを識別する重要性は極めて高いといえる．

設計段階でハードウェアトロイを検知するために，機械学習を用いた手法が提案されている．機械学習モデルとして，ニューラルネットワークを用いる手法 [4] や，ランダムフォレストを用いる手法 [5]，グラフ学習を用いる手法 [6] が存在する．中でも，グラフ学習を用いた識別手法 [6] は，従来の手法では表現できなかった特徴量抽出をグラフ畳み込みと呼ばれる操作によって実現する手法で，ハードウェアトロイ検知においても有効であることが確認されている．

一方で，グラフ学習を用いた識別器に対するメンバシップ推論攻撃の危険性が指摘されている [7], [8], [9]．メンバシップ推論攻撃とは，機械学習モデルにおけるプライバシー問題の一つであり，あるデータが識別器の学習データとして使われていたかどうかを推論する攻撃手法である．

グラフ学習を用いたハードウェアトロイ識別においてもメンバシップ推論攻撃の可能性がある．グラフ学習を用いたハードウェアトロイ識別に対するメンバシップ推論攻撃では，以下の 2 通りの攻撃方法がある．

- 通常回路のデータが，訓練データに含まれているかどうかを推論するメンバシップ推論攻撃
- 通常回路に挿入されたハードウェアトロイのデータが，訓練データに含まれているかどうかを推論するメンバシップ推論攻撃

特に，後者のハードウェアトロイに対するメンバシップ推論攻撃が成立する場合，攻撃者は攻撃対象となるハードウェアトロイ識別器の訓練に使われたハードウェアトロイの構造を入手できるため，訓練データに含まれていない構造のハードウェアトロイを設計情報に挿入することで攻撃対象の識別器が識別できないハードウェアトロイを含むネットリストを作成し，攻撃者は，検知不可能なハードウェアトロイの挿入に成功することになる．

本稿ではハードウェアトロイに対するメンバシップ推論攻撃に着目し，グラフ学習を用いたハードウェアトロイ識別器を対象としたメンバシップ推論攻撃を提案する．本稿はグラフ学習を用いたハードウェアトロイ識別器に対してメンバシップ推論攻撃を提案し，評価する初めての試みである．評価実験の結果，ハードウェアトロイのデータに関して，最大で AUC (Area Under the Curve) 0.988 の精度で攻撃が成功することを確認した．

本稿の貢献は以下の通りである．

- グラフ学習を用いたハードウェアトロイ識別器に対して，メンバシップ推論攻撃を提案した．提案手法は，攻撃対象と類似の振る舞いをする Shadow model を作成することで，訓練データに含まれるハードウェアトロイを与えたときの出力値と，訓練データに含まれないハードウェアトロイを与えたときの出力値を得る．そして，これらの出力値の差異を学習することで Attack model を作成し，攻撃対象のハードウェアトロイ識別器の訓練データに含まれるハードウェアトロイを識別する．
- グラフ学習を用いたハードウェアトロイ識別器に対するメンバシップ推論攻撃により，ハードウェアトロイを構成するノードが攻撃対象のハードウェアトロイ識別器の訓練データとして利用されたかどうかを最大で AUC 0.988 の精度で識別することに成功した．

本稿は以下のように構成される．2 章で，グラフ学習を用いたハードウェアトロイ識別手法を説明する．3 章で，本稿で提案するハードウェアトロイ識別器に対するメンバシップ推論攻撃手法を提案する．4 章で実験結果を説明する．5 章で本稿をまとめる．

2. グラフ学習を用いたハードウェアトロイ識別

本章では，[6] に基づき，グラフ学習を用いたハードウェアトロイ識別を紹介する．2.1 節でグラフ学習を説明し，2.2 節でグラフ学習を用いたハードウェアトロイ識別を説明する．

2.1 グラフ学習

グラフ学習とは，グラフ構造で表現されるデータを入力とした機械学習手法である [10]．グラフ構造は，各ノードに特徴ベクトルを付与し，ノード間の接続情報となるエッジに隣接行列を用いることで表現する．学習及び識別の際は，各ノードが自身と隣接するノードの特徴ベクトルを集約して自身の特徴ベクトルを更新する，グラフ畳み込みと呼

ばれる操作を適用する．集約する操作は AGGREGATE，更新する操作は COMBINE と呼ばれ，以下の式で表される [11]．

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\}) \quad (1)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)}) \quad (2)$$

式 (1) では，特徴ベクトルの更新対象とするノード v に隣接するノード群 u の特徴ベクトル $h_u^{(k-1)}$ を集約し，ベクトル $a_v^{(k)}$ を生成する．k はグラフ畳み込みの回数を表す．式 (2) において，生成された $a_v^{(k)}$ と，$k-1$ 回目の畳み込みまでで生成された更新対象ノード v の特徴ベクトル $h_v^{(k-1)}$ を用いて新たな特徴ベクトル $h_v^{(k)}$ を生成する．

一般的なグラフ学習では，グラフ畳み込みを複数回適用した後，全結合層を用いて識別結果を出力する．

2.2 ハードウェアトロイ識別手法

グラフ学習手法をハードウェアトロイ識別に適用する手法 [6] を説明する．

ハードウェアトロイ識別では，まず識別対象となる回路の設計情報をグラフ構造に変換する．回路の設計情報としてゲートレベルネットリストを利用する．ゲートレベルネットリストには，ネットリストを構成するゲート群と，ゲート間の接続情報となるネット群が記載されている．この情報を元に，各ゲートをノードに，各ネットをエッジに対応させることでグラフ構造に変換する．グラフ構造において，ハードウェアトロイを構成するノードをトロイノードと呼び，通常回路を構成するノードをノーマルノードと呼ぶ．

訓練の際は，作成したグラフの各ノードにトロイノードかノーマルノードかのラベルを付与し，識別器は各ノードのラベルを正しく推論できるように訓練する．識別の際は，入力されたグラフ構造の各ノードがトロイノードかノーマルノードかを予測する．

以上に述べたグラフ学習を用いたハードウェアトロイ識別の流れを図 1 に示す．我々が知る限り，グラフ学習を用いたハードウェアトロイ識別器 [6] (図 1 の Trained model) は，未知のハードウェアトロイの識別能力を含め，最も識別能力が高いものの 1 つである．本稿では，グラフ学習を用いたハードウェアトロイ識別器 [6] をメンバシップ推論攻撃の対象とする．

3. グラフ学習を用いたハードウェアトロイ識別に対するメンバシップ推論攻撃

本章では，グラフ学習を用いたハードウェアトロイ識別器のメンバシップ推論攻撃にあたり，まず攻撃者の条件をまとめる（3.1 節）．その後，グラフ学習を用いたハードウェアトロイ識別器のメンバシップ推論攻撃手法を提案する（3.2 節）．

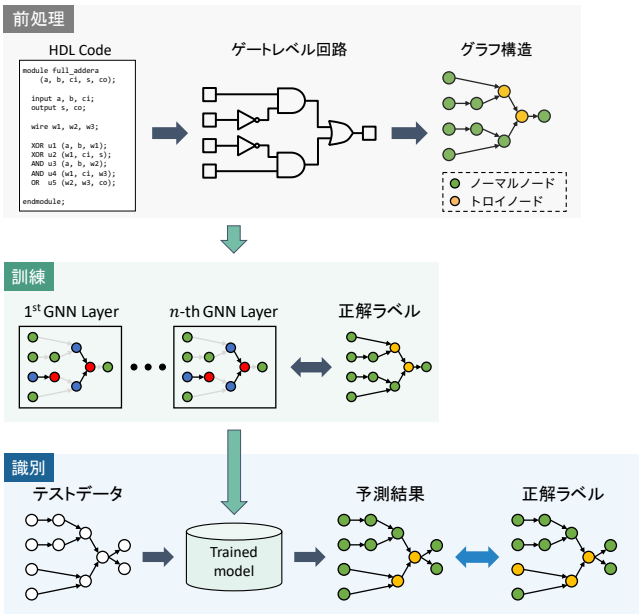


図 1: グラフ学習を用いたハードウェアトロイ識別の流れ．

3.1 攻撃者の条件

メンバシップ推論攻撃は，攻撃者が攻撃対象のグラフ学習モデルの情報をどれだけ入手できるかによって，攻撃の難易度が変化する．グラフ学習を用いたハードウェアトロイ識別に対するメンバシップ推論攻撃において，攻撃者に以下の条件を仮定する．

条件 1 攻撃者は，攻撃対象のグラフ学習モデルに使用されたグラフ学習手法及びハイパーパラメータを知っている．

条件 2 攻撃者は，攻撃対象のグラフ学習モデルに入力された特徴ベクトルの構成を知っている．

条件 3 攻撃者は，攻撃対象のグラフ学習モデルの訓練に使用されたネットリストを一部に含むネットリスト群を入手している．

グラフ学習を用いたハードウェアトロイ識別が一般的となり，多くのハードウェア検証ベンダがハードウェアトロイ識別を実施するようになれば，攻撃者にもその詳細情報が知られる恐れがあり，**条件 1** と**条件 2** が仮定できる．**条件 1** と**条件 2** より，攻撃者は攻撃対象のグラフ学習モデルと同じ構成で攻撃者側のモデルを作成できる．**条件 3** により攻撃者は攻撃対象のグラフ学習モデルの訓練に使用されたネットリストの一部を利用して，攻撃者側のモデルを作成できることになる．ベンチマークデータセットなど攻撃者もハードウェアトロイを知っている可能性があり，これに基づき**条件 3** を仮定している．

条件 1 から**条件 3** に加えて，以下の 2 つの条件を想定する．

条件 4 攻撃者は，攻撃対象のグラフ学習モデルの入出力に何度もアクセスできる．

条件 5 攻撃者は攻撃対象のグラフ学習モデルの出力値を得

る際に，確率に変換された値ではなく，確率に変換する前の最終層のグラフ畳み込みをした後の出力値を入手できる．

条件 4 が仮定できる場合，攻撃者は攻撃対象のグラフ学習モデルの出力値を教師ラベルとしたモデルの作成が可能となる．一方，**条件 5** が仮定できる場合は，予測確率に変換する前の予測値をもとに，メンバシップ推論攻撃が可能となる．**条件 4** と**条件 5** は，必ずしも攻撃者に仮定できるとは限らないが，以降，これらの条件が満足された際と満足されなかった際の双方の攻撃を検証する．

3.2 提案手法

本節では，[7], [8], [9] のもと，グラフ学習を用いたハードウェア識別に対するメンバシップ推論攻撃を提案する．提案手法の全体像を図 2 に示す．

3.2.1 Target model

まず前提として，グラフ学習を用いたハードウェアトロイ識別器を仮定し，これを攻撃対象とする．これは [6] に基づくものであり，このハードウェアトロイ識別器を Target model と呼ぶ．Target model の訓練に使われたネットリスト群を Target InTrain と呼び，使われなかったネットリスト群を Target OutTrain と呼ぶ．これらをまとめて，Target dataset と呼ぶ．

3.2.2 Shadow model

続いて，攻撃者は Target model に類似した動作を実現する Shadow model を準備する．Shadow model は Target model と等価ではないが，Target model と Shadow model とが類似した振る舞いをすることで，訓練データの特定につながる．

Shadow model の構成

メンバシップ推論攻撃を成功させるためには，Shadow model の入出力の挙動が Target model に近い方が好ましい．3.1 節で説明した条件を元に，攻撃者は以下のように Shadow model を作成する．

(1) ハードウェアトロイを含む複数のゲートレベルネットリストを準備する．
(2) 用意したゲートレベルネットリストの内，Shadow model の訓練に使用するネットリスト群を Shadow InTrain，訓練に使用しないネットリスト群を Shadow OutTrain とする．
(3) **条件 1** と**条件 2** より，Target model と同じ構成，すなわち，同一のグラフ学習モデルのもと，同一のハイパーパラメータと同一の特徴ベクトルを持つハードウェアトロイ識別器を，Shadow InTrain のみを用いて作成する．

更に，3.1 節の**条件 3** より，攻撃者は Target dataset に含まれるネットリストと同じネットリストを入力することも考えられる．上記の手順 (1) において，Target dataset に含まれるネットリストを多数用意すると，Shadow model の振る舞いはより Target model に近づくと考えられる．

教師ラベル：2 値ラベルと Target model の出力

Target model 及び Shadow model はどちらもグラフ学習を用いたハードウェアトロイ識別器であり，そのため，Shadow model を訓練する際の教師ラベルは，ネットリストに含まれる各ノードがハードウェアトロイを構成するノードかどうかの 2 値を与えることとなる．一方，Target model と Shadow model の振る舞いをより近くするため，

(1) Shadow InTrain を Target model に入力し，Target model から出力された値を調べ，
(2) その値を教師ラベルとして，Shadow model を作成する，

ことが考えられる．ただし，この手法を適用する場合は (1) を何度も実行する必要がある．すなわち，Target model によって Shadow InTrain に含まれるネットリストを何度も識別する必要がある．そこで，攻撃者に 3.1 節の**条件 4** が仮定できる場合，Target model の出力値を利用し Shadow model を構築するものとする．

3.2.3 Attack model

最後に，攻撃者は Attack model と呼ばれる識別器を作成する．一般的に，機械学習モデルは訓練に使用した既知のデータを予測する場合と，訓練に使用していない未知のデータを予測する場合の出力値には差異が生じる．攻撃者はこの性質を利用し，Target model の出力値を入力として，その差異をうまく識別する Attack model を作成する．

Attack model を訓練するために，Shadow model に訓練データならびにそれ以外のデータを入力し，その出力値を Attack model に入力する．Shadow model は攻撃者が作成したものであり訓練データは既知となるため，Attack model に対して，教師ラベルとして訓練データかそうでないかを与えることができる．そこで以下の手順で Attack model を作成する．

(1) 攻撃者は Shadow InTrain と Shadow OutTrain を Shadow model に入力し，その出力値を得る．
(2) 得られた出力値の中から，ハードウェアトロイを構成するノードのみを抽出する．Shadow InTrain に含まれるハードウェアトロイのノード集合を Shadow Trojan InTrain，Shadow OutTrain に含まれるハードウェアトロイのノード集合を Shadow Trojan OutTrain と呼ぶ．
(3) 攻撃者は全ての出力値について，訓練に使われたデータの出力値か，使われていないデータの出力値かを知っているため，訓練データかどうかを教師ラベルとして，Shadow Trojan InTrain に対する Shadow model の出力値と Shadow Trojan OutTrain に対する Shadow model の出力値を Attack model に入力し，Attack model を訓練する．

なお (1) において Shadow InTrain と Shadow OutTrain を Shadow model に入力し，その出力値を得る際，Shadow model の出力値として確率に変換された最終出力値そのものを使うものとする．これは通常，Target model の出力値が確率に変換された最終出力値そのものであるためである．

一方，**条件 5** が満足される場合，Target model から確率に変換する前の最終層のグラフ畳み込みをした後の出力値を得ることができる．その場合，Attack model を作成する際に，Shadow model の出力値として，確率に変換する前の最終層のグラフ畳み込みをした後の出力値を用いることにする．

3.2.4 メンバシップ推論攻撃

Attack model が作成された後，攻撃者は，Target model に対し以下の手順でメンバシップ推論攻撃を実施する．

(1) 攻撃者は，攻撃対象のネットリストを準備する．ネットリストはハードウェアトロイが含まれるものとし，攻撃者はトロイノードがどれかを知っているものとする．なお攻撃者は，この時点では攻撃対象ネットリストが Target model の訓練に利用されたかどうかは知らない．

(2) 攻撃対象ネットリストを Target model に入力し，攻撃対象ネットリストを構成するトロイノードから得られた出力値を得る．

(3) トロイノードの出力値を Attack model に入力し，Attack model は入力されたデータが Target model の訓練データに含まれるかどうかを予測する．

以上の手順により，攻撃者は攻撃対象ネットリストのハードウェアトロイが Target model で学習済みかどうかを知ることができる．

4. 評価実験

本章では，3.1 節で説明した通り，攻撃者が**条件 1** から**条件 3** を満たしている上で，**条件 4** と**条件 5** を満たす場合と満たさない場合のメンバシップ推論攻撃の精度を評価する．

4.1 節で Target detaset と Shadow dataset の作成方法を説明する．4.2 節で Shadow dataset のネットリスト選択方法を説明する．4.3 で実験条件を説明する．4.4 節で Attack model の識別結果を説明する．

4.1 Target detaset と Shadow dataset の作成

実験をするにあたり，まず Target dataset，及び Shadow dataset を作成する．ハードウェアトロイ識別器に対するメンバシップ推論攻撃では，訓練データにおけるハードウェアトロイのデータを識別器に入力した際と，訓練データ以外のハードウェアトロイのデータを識別器に入力した際の出力値の差異を学習する．そのため，実験する際はハードウェアトロイ識別器の訓練データに含まれたネットリストとそれ以外のネットリストを区別する必要がある．

表 1: InTrain/OutTrain database の構成.

InTrain database		OutTrain database	
名前	Netlist	名前	Netlist
A1	B19-T100	B1	MAC10GE-T700
A2	B19-T200	B2	MAC10GE-T710
A3	s35932-T100	B3	MAC10GE-T720
A4	s35932-T200	B4	MAC10GE-T730
A5	s35932-T300	B5	RS232-T1000
A6	s38417-T100	B6	RS232-T1100
A7	s38417-T200	B7	RS232-T1200
A8	s38417-T300	B8	RS232-T1300
A9	s38584-T100	B9	RS232-T1400
A10	s38584-T200	B10	RS232-T1500
A11	s38584-T300	B11	RS232-T1600
A12	wb_conmax	B12	s15850-T100

以上を考慮し，表 1 に示す InTrain/OutTrain database を用意する．InTrain/OutTrain database には Trust-HUB [12], [13] に公開されている 24 種類のネットリストを使用する．24 種類のネットリストを，“InTrain database” に 12 種類，“OutTrain database” に 12 種類のネットリストに分ける．Target InTrain，及び Shadow InTrain を作成する際は，表 1 における “InTrain database” のみからネットリストを選択し，Target OutTrain，及び Shadow OutTrain を作成する際は，表 1 における “OutTrain database” のみからネットリストを選択する．これにより，訓練データのネットリストとそうでないネットリストが重複しないデータセットを作成する．

なお Target dataset と Shadow dataset は一部重複があり得る．4.2 節に示すように Target dataset と Shadow dataset との間で重複が大きい場合と，重複が全くない場合と複数通り考えることにする．

4.2 Shadow dataset のネットリスト選択方法

本節では，Shadow dataset に使用するネットリストの選択方法を説明する．

本章の実験では，Shadow dataset と Target dataset のネットリストの重複率とメンバシップ推論攻撃の精度の関係性を確認するため，表 2 に示すように Target dataset のネットリストの構成から少しずつ変化させた 7 種類の Shadow dataset を用意する．表 2 の “構成” に書かれたネットリストの名前は表 1 における “名前” と対応している．

表 2 の “重複率” は，Shadow dataset の構成に含まれる 12 個のネットリストの内，Target dataset の構成に含まれる 12 個のネットリストといくつのネットリストが重複しているかを示す．重複率が高いほど Shadow dataset の構成は Target dataset に近づくため，メンバシップ推論攻撃

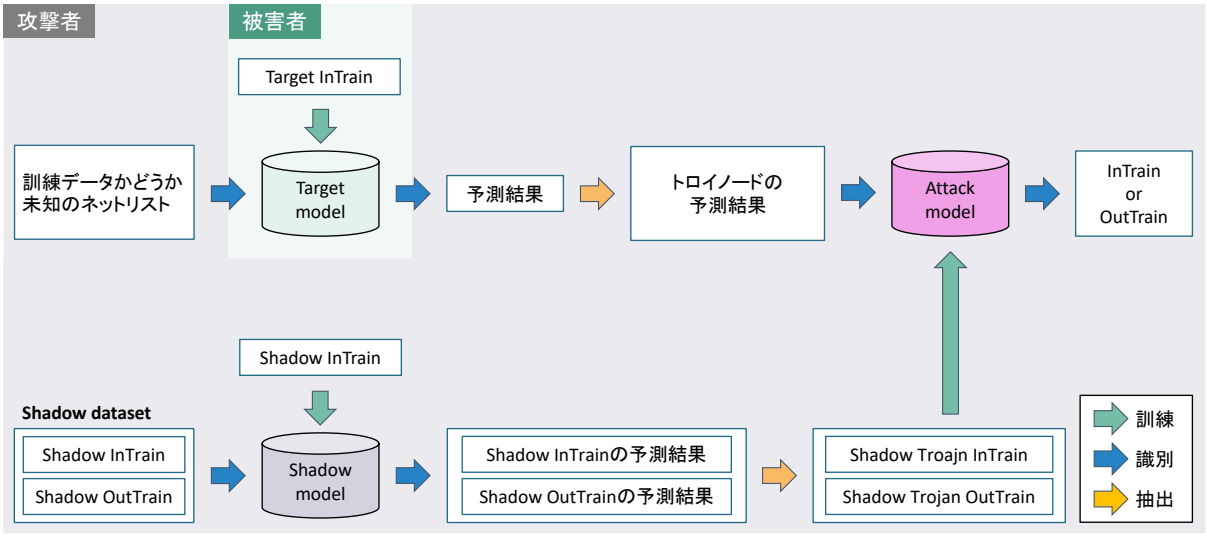


図 2: グラフ学習を用いたハードウェアトロイ識別器に対するメンバシップ推論攻撃の流れ.

表 2: 実験に使用する Target dataset 及び Shadow dataset の構成と，Shadow dataset に含まれるネットリストの内，何個が Target dataset に含まれるネットリストと同一かの重複率.

dataset	No.	重複率	InTrain/OutTrain	構成
Target	-	-	InTrain	A1, A3, A6, A9, A10, A12
			OutTrain	B1, B3, B5, B7, B9, B12
Shadow	⓿	$\frac{12}{12}$	InTrain	A1, A3, A6, A9, A10, A12
			OutTrain	B1, B3, B5, B7, B9, B12
	❶	$\frac{10}{12}$	InTrain	A1, A3, A6, A9, A10, A11
			OutTrain	B1, B3, B5, B7, B9, B11
	❷	$\frac{8}{12}$	InTrain	A1, A3, A6, A9, A8, A11
			OutTrain	B1, B3, B5, B7, B10, B11
	❸	$\frac{6}{12}$	InTrain	A1, A3, A6, A7, A8, A11
			OutTrain	B1, B3, B5, B8, B10, B11
	❹	$\frac{4}{12}$	InTrain	A1, A3, A5, A7, A8, A11
			OutTrain	B1, B3, B6, B8, B10, B11
	❺	$\frac{2}{12}$	InTrain	A1, A4, A5, A7, A8, A11
			OutTrain	B1, B4, B6, B8, B10, B11
	❻	$\frac{0}{12}$	InTrain	A2, A4, A5, A7, A8, A11
			OutTrain	B2, B4, B6, B8, B10, B11

表 3: optuna に指定するパラメータと探索範囲.

パラメータ名	変化する値	探索範囲	
		下限	上限
eta	学習率	0.001	0.5
gamma	損失減少の下限値	0.0001	1
max_depth	決定木の深さの最大値	4	30
min_child_weight	決定木の葉の重みの下限	1	10
subsample	各決定木において抽出されるデータの割合	0.1	1
lambda	L2 正則化項	0.01	1
alpha	L1 正則化項	0.01	1

表 4: Target model 及び Shadow model に入力される特徴ベクトル.

#	説明
1	プライマリ入力までの距離
2	プライマリ出力までの距離
3	入力側の信号線数
4	出力側の信号線数
5	論理ゲート (0 出力確率)
6	論理ゲート (1 出力確率)
7–46	ノード種別

が成功しやすくなると考えられる.

4.3 実験条件

表 2 に示すデータセットを用いる．各 Shadow dataset に対応する Attack model を作成し，Target dataset を Target model に入力した際の出力値に対してメンバシップ推論攻撃を適用することで Attack model の性能を評価する．Attack model には XGBoost [14] を用いる.

Attack model のパラメータ探索には optuna [15] を用いる．optuna は，パラメータ探索にベイズ最適化を用いるフレームワークである．通常パラメータ探索にはグリッドサーチやランダムサーチが用いられる．グリッドサーチは考え得るパラメータの組み合わせ全てを網羅的に探索する手法，ランダムサーチは考え得るパラメータの内，いくつかをランダムに選択し，探索する手法である．グリッドサーチは最適なパラメータを見つけるためにかかる時間が膨大である点，ランダムサーチは最適なパラメータを見つけることが困難な点が問題とされている．一方，optuna はパラメータ探索にベイズ最適化の原理を用いることで，過去に設定したパラメータと，その時に得られた評価関数の値を基に効率的に最適なパラメータを探索することができるため，グリッドサーチより短い時間で，ランダムサーチより確実に最適なパラメータを探索できる．

optuna を適用する際は，探索させるパラメータの種類と探索範囲，及び探索の指標となる評価関数を指定する．本稿の実験で optuna に指定するパラメータを表 3 に示す．評価関数には AUC を指定する．表 3 において，max_depth と min_child_weight の optuna の探索空間は整数であり，その他のパラメータの探索空間は連続値である．

実験の際は optuna によるパラメータ探索を 50 回実施し，AUC が最も高かった際の識別器を Attack model として採用する．

使用する Target model 及び shadow model は，3.1 節で説明した**条件 1**，**条件 2** より，使用するグラフ学習手法，パラメータ及び特徴ベクトルは同じとする．使用するグラフ学習手法は MPNN [16] で，畳み込みの層数は 2 層とする．入力のユニット数は特徴ベクトルの大きさと対応する 46 個で，中間層のユニット数が 32 個，出力層のユニット数を 2 個とする．各ノードに付与した特徴ベクトルを表 4 に示す．

4.4 Attack model の識別結果

本節では，4.2 節で説明した 7 種類の Shadow dataset を用いて Attack model を作成し，メンバシップ推論攻撃の精度を AUC により評価する．

Attack model のテストデータの用意及び識別は以下の手順に従う．

(1) 表 2 の Target dataset の全てのネットリストを Target model に入力し，出力値を得る．
(2) 得られた出力値の中から，トロイノードのみを抽出する．Target InTrain に含まれるハードウェアトロイのノード集合を Target Trojan InTrain，Target OutTrain に含まれるハードウェアトロイのノード集合を Target Trojan OutTrain と呼ぶ．
(3) Target Trojan InTrain と Target Trojan OutTrain に含まれるノードをテストデータとして，Attack model で識別する．Target Trojan InTrain に含まれる各ノードを訓練データと識別した場合を True Positive，Target Trojan InTrain に含まれる各ノードを訓練データでないと識別した場合を False Negative，Target Trojan OutTrain に含まれる各ノードを訓練データと識別した場合を False Positive，Target Trojan InTrain に含まれる各ノードを訓練データでないと識別した場合を True Negative とした．

評価する際は，3.1 節における**条件 4** を満たし，Target model の出力値を教師ラベルとした Shadow model の作成が可能な場合とそうでない場合，**条件 5** を満たし，予測確率に変換する前の予測値を出力とする Shadow model の作成が可能な場合とそうでない場合の合計 4 種類の手法の識別結果を比較する．

実験結果を表 5 に示す．表 5 より，全ての Shadow dataset において AUC が 0.8 を下回ることがなく，Target dataset とのネットリストの重複率に関わらずメンバシップ推論攻撃が成功することが確認できた．

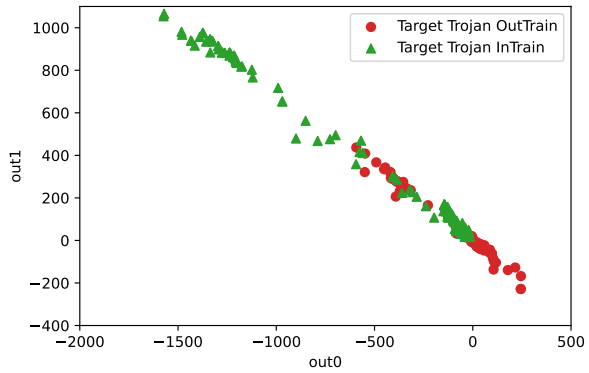
また，4 種類の手法の中で Attack model の精度が最も高くなったのは，**条件 4** が満たされず，**条件 5** が満たされている場合であった．そのため，グラフ学習を用いたハードウェアトロイ識別器に対するメンバシップ推論攻撃を成功させるためには，**条件 4** は満たす必要のない条件であることが分かる．Target model に対して何度もアクセスすることは現実的に困難であるため，メンバシップ推論攻撃を適用する上では望ましい結果だといえる．

条件 5 が満たされない場合に識別精度が低下する理由を考察する．本稿の実験に使用した Target dataset は，Target Trojan InTrain のデータ数が 217 個，Target Trojan OutTrain のデータ数が 88 個で構成されており，確率に変換される前の出力値は図 3 (a) のように分布している．対して，確率に変換した後のデータは図 3 (b) のように分布している．

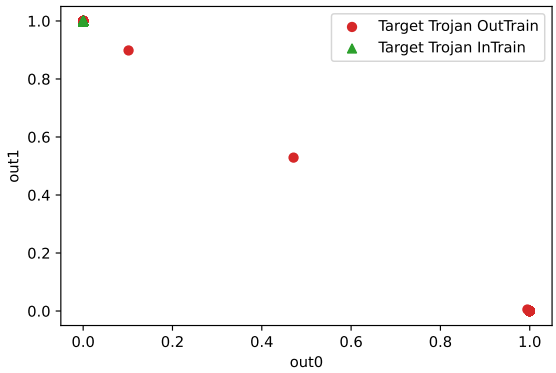
図 3 (a) のデータは，InTrain と OutTrain 間で重複しているデータが 1 つも存在しないが，図 3 (b) のデータは，6 個の Target Trojan OutTrain が Target Trojan InTrain と重複している．図 3 (b) より，確率に変換された Target Trojan InTrain は全て同じ特徴ベクトルに変換されていることが分かるため，確率に変換された後のデータでは，Target Trojan OutTrain は最大で 82 個のデータしか正しく識別できないことが分かる．グラフ学習を用いたハードウェアトロイ識別器に対するメンバシップ推論攻撃には，**条件 5** が満たされた方が良い．

5. おわりに

本稿では，グラフ学習を用いたハードウェアトロイ識別器に対してメンバシップ推論攻撃手法を提案し，攻撃の有効性を検証した．検証の結果，Target model の訓練に使用



(a) Target Trojan InTrain と Target Trojan OutTrain を Target model で予測したデータの分布.



(b) 図 3 (a) の出力値を確率に変換したデータの分布.

図 3: Target Trojan InTrain と Target Trojan OutTrain のデータの分布.

表 5: **条件 4** と**条件 5** が満たされる場合とそうでない場合における，各 Shadow dataset を用いて作成した Attack model の識別結果.

条件 4	条件 5	Shadow dataset						
		⓿	❶	❷	❸	❹	❺	❻
○	○	0.984	0.826	**0.896**	0.842	**0.870**	0.847	**0.847**
	×	0.865	0.836	0.841	0.846	0.836	0.847	0.841
×	○	**0.988**	**0.847**	0.833	**0.847**	0.847	**0.852**	**0.847**
	×	0.853	**0.847**	0.833	0.838	0.841	0.847	0.830

されたハードウェアトロイのデータを，最大で AUC 0.988 の精度で識別できることを確認した．今後の課題として，ハードウェアトロイの識別性能を落とさず，メンバシップ推論攻撃に対して堅牢性が高い識別器を作成することが挙げられる．

参考文献

[1] Rostami, M., Koushanfar, F., Rajendran, J. and Karri, R.: Hardware security: Threat models and metrics, *Proceedings of 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, pp. 819–823 (2013).

[2] Francq, J. and Frick, F.: Introduction to hardware Trojan detection methods, *Proceedings of 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 770–775 (2015).

[3] Xiao, K., Forte, D., Jin, Y., Karri, R., Bhunia, S. and Tehranipoor, M.: Hardware trojans: Lessons learned after one decade of research, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 22, No. 1, pp. 1–23 (2016).

[4] Hasegawa, K., Yanagisawa, M. and Togawa, N.: Empirical Evaluation and Optimization of Hardware-Trojan Classification for Gate-Level Netlists Based on Multi-Layer Neural Networks, *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 101, No. 12, pp. 2320–2326 (2018).

[5] Hasegawa, K., Yanagisawa, M. and Togawa, N.: Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier, *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, pp. 1–4 (2017).

[6] Hasegawa, K., Yamashita, K., Hidano, S., Fukushima, K., Hashimoto, K. and Togawa, N.: Node-wise Hardware Trojan Detection Based on Graph Learning, *arXiv preprint arXiv:2112.02213* (2021).

[7] Olatunji, I. E., Nejdl, W. and Khosla, M.: Membership inference attack on graph neural networks, *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, IEEE, pp. 11–20 (2021).

[8] He, X., Wen, R., Wu, Y., Backes, M., Shen, Y. and Zhang, Y.: Node-level membership inference attacks against graph neural networks, *arXiv preprint arXiv:2102.05429* (2021).

[9] Zhang, Z., Chen, M., Backes, M., Shen, Y. and Zhang, Y.: Inference attacks against graph neural networks, *Proceedings of the 31th USENIX Security Symposium*, pp. 1–18 (2022).

[10] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G.: The graph neural network model, *IEEE transactions on neural networks*, Vol. 20, No. 1, pp. 61–80 (2008).

[11] Xu, K., Hu, W., Leskovec, J. and Jegelka, S.: How powerful are graph neural networks?, *arXiv preprint arXiv:1810.00826* (2018).

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

[13] Salmani, H., Tehranipoor, M. and Karri, R.: On design vulnerability analysis and trust benchmarks development, *2013 IEEE 31st international conference on computer design (ICCD)*, IEEE, pp. 471–474 (2013).

[14] Chen, T. and Guestrin, C.: Xgboost: A scalable tree boosting system, *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794 (2016).

[15] Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M.: Optuna: A next-generation hyperparameter optimization framework, *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631 (2019).

[16] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. and Dahl, G. E.: Neural message passing for quantum chemistry, *International conference on machine learning*, PMLR, pp. 1263–1272 (2017).