

# 実用的な深層学習に対する Model Reverse-Engineering 攻撃の脅威評価

中井 綱人<sup>1,a)</sup> 鈴木 大輔<sup>1</sup> 大松 史生<sup>1</sup> 藤野 毅<sup>2</sup>

**概要:** 深層学習を活用した AI(Artificial Intelligence) の発展が目覚ましく、リアルタイム処理などを目的に、エッジ AI デバイスが注目されている。一方で、エッジ AI デバイスから発生する消費電力や漏洩電磁波等のサイドチャネル情報を用いて、学習モデル情報を窃取する Model Reverse-Engineering 攻撃が 2018 年から研究されはじめている。先行研究では、漏洩電磁波を用いて、実装が既知で単純なニューラルネットワークに対する Model Reverse-Engineering 攻撃の脅威を報告している。現実的な攻撃シナリオを考えた場合、より実用的で複雑なニューラルネットワークに対する脅威評価や、攻撃者の前提知識の整理が必要であり、先行研究では明らかにされていない。そこで、本論文では、マイコン向け深層学習開発プラットフォーム uTensor で開発された実用的なニューラルネットワークを対象に、Model Reverse-Engineering 攻撃の脅威を評価した。実験結果により、Model Reverse-Engineering 攻撃は、実用的なニューラルネットワークにおいても、ある程度現実的な脅威であることを示す。

**キーワード:** 深層学習, Model Reverse-Engineering 攻撃, サイドチャネル攻撃, Model Extraction 攻撃

## Model Reverse-Engineering Attack against Practical Deep Neural Network

TSUNATO NAKAI<sup>1,a)</sup> DAISUKE SUZUKI<sup>1</sup> FUMIO OMATSU<sup>1</sup> TAKESHI FUJINO<sup>2</sup>

**Abstract:** Artificial intelligence (AI), especially deep learning (DL), has been remarkable and applied to various industries. Moreover, edge AI devices are attracting attention because of real time processing. However, model reverse-engineering attack, which reveal model architecture and parameters utilizing side channel information such as power consumption or electromagnetic radiation, has been studied since 2018. The previous work has reported that model architecture and parameters of simple neural network whose implementation is known were revealed utilizing electromagnetic radiation. Considering of practical attack scenario, it is not clear that the threat in case of more complex and practical deep neural network and the prior knowledge of attackers in previous works. Therefore, this paper shows the evaluation of model reverse-engineering attack against the practical deep neural network which was developed on uTensor platform for microcontroller unit (MCU). According to the experimental result, model reverse-engineering attack becomes a certain threat against practical deep neural networks.

**Keywords:** Deep learning, Model reverse-engineering attack, Side channel attack, Model Extraction attack

### 1. はじめに

深層学習を活用した AI は、画像認識、自然言語処理、音声認識などで、驚異的な成果を上げている。そのため、深層学習を用いたアプリケーションやシステム、デバイス

<sup>1</sup> 三菱電機株式会社 情報技術総合研究所  
Information Technology R&D Center, Mitsubishi Electric Corporation

<sup>2</sup> 立命館大学 理工学部  
College of Science and Engineering, Ritsumeikan University

<sup>a)</sup> Nakai.Tsunato@dy.MitsubishiElectric.co.jp

の数は増加しており、様々な産業への適用が期待されている。例えば、自動車産業では、物体認識などに深層学習を用いて、自動運転のテストを行っている。顔認証システムでは、監視カメラやセキュリティゲート、スマートフォンのロック解除に、深層学習が用いられている。

また、深層学習のリアルタイム処理や分散処理を目的に、IoT(Internet of Things) 機器あるいは IoT 機器に近い場所で学習、推論を行うエッジ AI デバイスが注目されている。エッジ AI デバイスでは、処理速度や消費電力、精度、価格等の要件に応じて、MCU(Microcontroller Unit)、FPGA(Field-Programmable Gate Array)、組込み向け GPU(Graphics Processing Unit)、ASIC(Application Specific Integrated Circuit)、といったさまざまな実装が検討されている。

深層学習が浸透するに伴って、深層学習に対するセキュリティやプライバシー保護の課題が指摘されている。特に、学習モデル情報(学習モデルのアーキテクチャや大量の学習データを用いてチューニングされた各パラメータ)の窃取は、重要なセキュリティ課題の1つと言える。学習モデル情報は、学習モデルの作成者のノウハウとなる部分が多い。また、学習モデル情報が明らかな場合、入力データに僅かなく乱情報を加えることで深層学習に誤判断を引き起こす Adversarial Examples 攻撃 [1] が、強力かつ容易に実行できる。

学習モデル情報の窃取に関する攻撃は、Model Extraction 攻撃と Model Reverse-Engineering 攻撃に大別できる。Model Extraction 攻撃は、2016年に Tramèr らによって提案された攻撃手法である [2]。攻撃対象は、クラウド上に実装された深層学習システムで、API が公開されている想定である。攻撃者は、攻撃対象の深層学習システムから少ないクエリ数とその結果を用いて、同等、もしくは、より高い精度の代替モデルを作成する。ここで、代替モデルとは、攻撃者の手元で作成された学習モデルであり、元の学習モデルと学習モデル情報は異なるが、性質は同等であるモデルのことを指す。また、Tramèr らは、Model Extraction 攻撃の対策として、出力する確率情報を丸めること (Rounding confidences) が効果的であると述べている。一方で、Model Reverse-Engineering 攻撃は、攻撃対象の学習モデル情報を窃取する攻撃手法である。攻撃対象は、AI エッジデバイスで、そこから発生する消費電力や処理時間といったサイドチャンネル情報を用いる。攻撃者は、代替モデルではなく、元の学習モデルと学習モデル情報が一致する複製モデルの作成を目的とする。2018年に Batina らは、MCU 実装の簡単なニューラルネットワークに対して、漏洩電磁波からアーキテクチャだけでなく、パラメータ、活性化関数の種類がリークすることを示した [3]。また、Model Reverse-Engineering 攻撃は、Model Extraction 攻撃の対策 Rounding confidences を無効化できるため、非

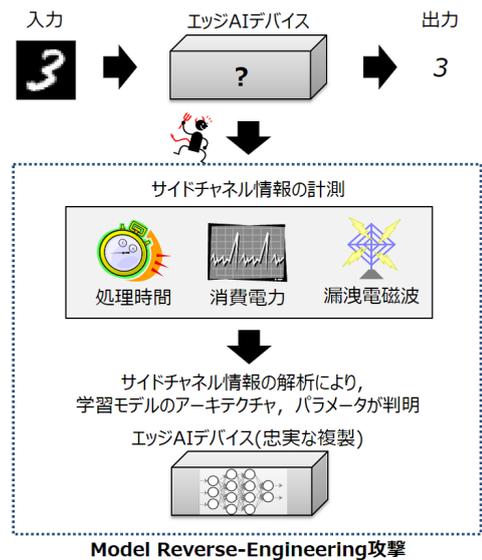


図 1 Model Reverse-Engineering 攻撃における攻撃シナリオ

常に脅威と言える。

本論文では、実用的なニューラルネットワークを対象に、Model Reverse-Engineering 攻撃の脅威を評価する。Batina らの攻撃 [3] は、MCU 実装の単純なニューラルネットワーク (6 入力, 5 出力, 3 層) を対象としている。また、実装が既知である前提での攻撃である。より現実的な攻撃シナリオを考えた場合、複雑なニューラルネットワークに対する脅威評価や、攻撃者の前提知識の整理が必要であり、先行研究では明らかにされていない。そこで、本論文では、MCU 向け深層学習開発プラットフォーム uTensor[4] で開発された手書き文字識別用ニューラルネットワークを対象に、漏洩電磁波を用いた Model Reverse-Engineering 攻撃を実施した。また、攻撃実施にあたり、攻撃者の前提知識がどれくらい必要か、攻撃の手順を整理し、また、攻撃コストを考察した。

本論文の貢献は、uTensor で開発された実用的な MCU 実装のニューラルネットワークに対して、初めて Model Reverse-Engineering 攻撃を実施した点にある。そして、攻撃実施にあたり、攻撃者の前提知識、攻撃手順、攻撃コスト、を整理・考察した点にある。本論文により、実用的な MCU 実装のニューラルネットワークにおいても、Model Reverse-Engineering 攻撃の脅威がある程度現実的であることを示す。

## 2. 準備

本章では、Model Reverse-Engineering 攻撃における、攻撃シナリオと窃取する学習モデル情報の整理、Model Extraction 攻撃との違い、先行研究の課題について示す。

### 2.1 攻撃シナリオ

Model Reverse-Engineering 攻撃における攻撃シナリオ

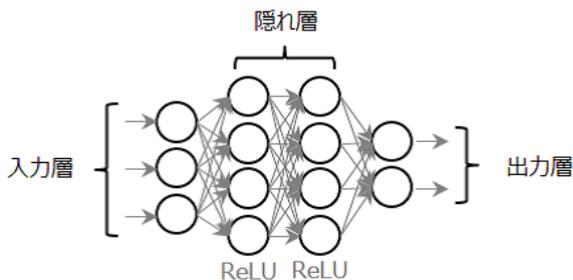


図 2 3層 MLP のアーキテクチャ例

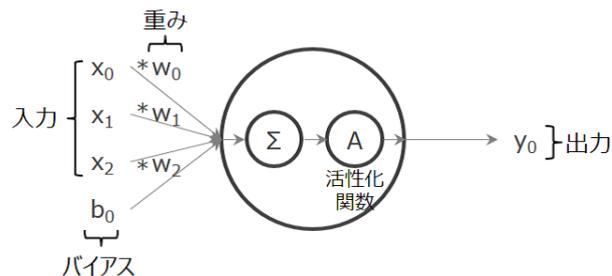


図 3 パーセプトロンの処理

を図 1 に示す。攻撃対象は、エッジ AI デバイスを想定する。攻撃者の目的は、エッジ AI デバイスの学習モデル情報（アーキテクチャ、パラメータ）を手に入れることである。

前提条件として、エッジ AI デバイスにおける学習モデル情報は、機密情報として、セキュアに格納されているとする。つまり、エッジ AI デバイスはブラックボックスである。攻撃者は、エッジ AI デバイスへ入力データを与え、出力結果を受け取ることはできる。

Model Reverse-Engineering 攻撃では、攻撃者は、エッジ AI デバイスが動作しているときの、処理時間、消費電力、漏洩電磁波といったサイドチャンネル情報を計測する。そして、計測したサイドチャンネル情報を解析することで、学習モデル情報を抽出する。

## 2.2 窃取する学習モデル情報

Model Reverse-Engineering 攻撃で窃取する学習モデル情報についてまとめる。学習モデル情報は、学習モデルのアーキテクチャと各パラメータに大別できる。

深層学習における学習モデルのアーキテクチャとは、層の数とそれを構成するパーセプトロンの数、活性化関数の種類である。また、CNN の場合は、畳み込み演算の構成も学習モデルのアーキテクチャである。枝刈りがある場合、各パーセプトロンの結線情報も学習モデルのアーキテクチャと言える。MLP(Multilayer perceptron) の場合は、各パーセプトロンが全結合であるため、結線情報は自明である。図 2 に 3 層 MLP の例を示す。図 2 の学習モデルのアーキテクチャは、層数が 3、パーセプトロン数が 13、活性化関数が ReLU 関数である。

深層学習における各パラメータとは、各パーセプトロンにおける重みとバイアスである。図 3 にパーセプトロンの処理を示す。図 3 では、重みを  $w$ 、バイアスを  $b$  で示している。これらパラメータは、大量の学習データを用いて、チューニングされる。

## 2.3 Model Extraction 攻撃との差分

Model Reverse-Engineering 攻撃と Model Extraction 攻撃の違いを表 1 に示す。また、参考として、学習モデルの圧縮・蒸留との比較も示している。学習モデルの圧縮・蒸

留は、学習モデルの計算コスト/学習モデルサイズを低減するために、学習モデル情報を用いて、同等な性能かつ、コンパクトな学習モデルを生成する技術である。

Model Reverse-Engineering 攻撃と Model Extraction 攻撃は、攻撃で得られる情報（出力）が異なる。Model Reverse-Engineering 攻撃の目的は、攻撃対象の学習モデルの複製である。したがって、攻撃で得られる出力は、複製モデルとなる。一方で、Model Extraction 攻撃の目的は、攻撃対象の学習モデルと同等な学習モデルを入手することにある。したがって、学習モデル情報は、攻撃対象と必ずしも一致しない。攻撃で得られる出力は、代替モデルとなる。

両攻撃とも、Adversarial Examples 攻撃の前段階としての情報収集も目的として考えられる。Adversarial Examples 攻撃は、学習モデル情報を用いることで強力な攻撃が可能となる（ホワイトボックス攻撃）[1], [5]。Model Extraction 攻撃の場合は、代替モデルの学習モデル情報から Adversarial Examples を生成する。この攻撃の効果は、学習モデルの転移性に依存する。転移性とは、深層学習の特性の 1 つで、ある学習モデルが、同様なタスクを実行する別の学習モデルにおいても同じような性質を持つことである。Papernot らは、Model Extraction 攻撃によって得られた代替モデルを用いて、Adversarial Examples 攻撃に成功している [6]。しかし、必ずしも転移性が保証されているわけではない。Model Reverse-Engineering 攻撃の場合は、複製モデルの学習モデル情報から Adversarial Examples を生成するため、転移性に依存しない。したがって、Model Reverse-Engineering 攻撃による Adversarial Examples 攻撃は、より強力であると言える。

Model Extraction 攻撃の対策として、出力する確率情報を丸める Rounding confidences[2] が有効である。しかし、確率情報を用いない Model Reverse-Engineering 攻撃には、無効である。Model Reverse-Engineering 攻撃の対策としては、従来の暗号実装で用いられてきたサイドチャンネル対策の適用が考えられるが、実装評価までは至っていない [3]。

## 2.4 先行研究の課題

Model Reverse-Engineering 攻撃の先行研究は、2018 年

表 1 Model Reverse-Engineering 攻撃と Model Extraction 攻撃の比較

|    | Model Reverse-Engineering 攻撃           | Model Extraction 攻撃                       | (参考) 蒸留・圧縮         |
|----|--|---|--------------------|
| 目的 | ・学習モデルの複製<br>・Adversarial Examples の生成 | ・同等な学習モデルの入手<br>・Adversarial Examples の生成 | ・計算コスト/学習モデルサイズの低減 |
| 出力 | ・複製モデル                                 | ・代替モデル                                    | ・コンパクトな学習モデル       |
| 条件 | ・入出力データのみ 既知                           | ・入出力データのみ既知                               | ・学習モデル情報も既知        |
| 要因 | ・サイドチャネルリーク                            | ・転移性                                      | ・転移性               |
| 対策 | ・サイドチャネル対策の適用 (未検証)                    | ・Rounding confidences(API 制限) 対策          | - (攻撃ではないため)       |

に Hua らが, FPGA 実装の CNN アクセラレータにおいて, メモリアクセスのパターン (メモリアドレス/時間) から CNN のアーキテクチャがリークすることを示したことから始まる [7]. また, Hua らは, Zero pruning 処理を用いた CNN アクセラレータの場合, メモリアクセスのパターンから CNN の重みがリークすることを明らかにした. 同様に, キャッシュベースのサイドチャネル攻撃がいくつか報告されている [8], [9], [10]. また, Duddu らは, 処理時間の傾向をプロファイリングすることで, 学習モデルのアーキテクチャを明らかにする攻撃を提案した [11].

一方で, 消費電力や漏洩電磁波を用いた Model Reverse-Engineering 攻撃は, より強力である. 2018 年に Batina らは, MCU 実装のニューラルネットワークを対象に, 漏洩電磁波からアーキテクチャだけでなく, パラメータ, 活性化関数の種類がリークすることを明らかにした [3]. Batina らの攻撃は, これまでの Model Reverse-Engineering 攻撃 [7], [8], [9], [10], [11] の中で, 最も学習モデル情報が窃取できている. また, Yoshida らは, FPGA 実装の DNN (Deep Neural Network) アクセラレータを対象に, 漏洩電磁波を解析することで, 重みを明らかにした [12]. Xiang らは, Raspberry Pi 上に実装した DNN を対象に, 消費電力傾向をプロファイリングすることで, 学習モデルのアーキテクチャを明らかにする攻撃を提案した [13].

消費電力や漏洩電磁波を用いた Model Reverse-Engineering 攻撃において, より現実的な攻撃シナリオを考えた場合, より実用的で複雑なニューラルネットワークに対する脅威評価や, 攻撃者の前提知識の整理が必要である. 先行研究では, 独自の実装でかつ, 単純なニューラルネットワークに対するものが多い. 例えば, Batina らの攻撃は, 実装が既知な MCU 実装の単純なニューラルネットワーク (6 入力, 5 出力, 3 層) を対象としている. したがって, 先行研究では明らかにされていない実装が不明なブラックボックスなエッジ AI デバイスに対する攻撃の手順や攻撃者の前提知識の整理が必要である.

### 3. 評価環境

本章では, 評価環境について, 開発環境, 学習モデル, 攻撃評価環境, を示す. 手書き文字識別用の学習モデルを uTensor[4] で開発し, 漏洩電磁波を用いた Model Reverse-

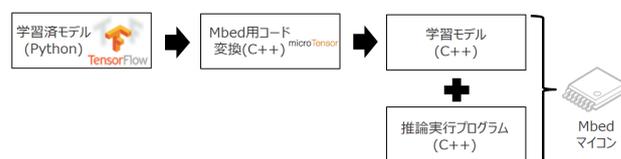


図 4 uTensor を用いた開発フローの概略

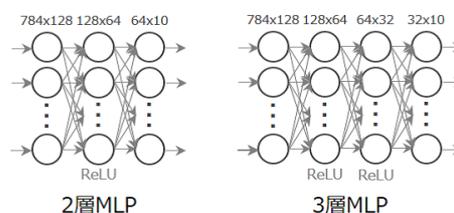


図 5 実装した MNIST 識別用の学習モデル

Engineering 攻撃の評価環境を構築した.

#### 3.1 開発環境

開発環境は, MCU 向け深層学習開発プラットフォーム uTensor[4] を用いた. uTensor とは, ARM 社のマイコンボードおよびそのプログラミング環境である Mbed[14] と, デフォルトな学習フレームワーク TensorFlow[15] 上に構築された非常に軽量な深層学習用の推論フレームワークである. その有用性から, uTensor は, TensorFlow との統合が進んでいる.

図 4 に, uTensor を用いた開発フローの概略を示す. TensorFlow 上で開発した学習済モデルを, uTensor に入力することで, 学習モデルが Mbed 用のコードに変換される. 次に, 学習モデルを実行させる推論実行プログラムを作成する. 最後に, Mbed 対応の MCU へ, 学習モデルのコードと, 推論実行プログラムを転送することで開発ができる. したがって, Python+TensorFlow で開発した学習モデルを EndToEnd で MCU に実装できる. ただし, 対応している TensorFlow の関数は, 限定される [4].

#### 3.2 学習モデル

実装した学習モデルは, 手書き数字を識別する 2 層 MLP と 3 層 MLP である. 図 5 に実装した学習モデルを示す. 表 2 に, 実装した学習モデルにおける学習モデル情報をまとめた. 学習データは, 手書き数字サンプル MNIST で, 20,000 サンプルを用いた. 推論時の入力データは, MNIST

表 2 実装した学習モデルにおける学習モデル情報

|          | 2層 MLP  | 3層 MLP  |
|----------|---------|---------|
| 層数       | 2       | 3       |
| 活性化関数    | ReLU    | ReLU    |
| パーセプトロン数 | 202     | 234     |
| 重み数      | 109,184 | 110,912 |
| バイアス数    | 202     | 234     |

データで、float32 型の 784 個のデータ列である。出力データは、数字 0~10 の内、最もスコアが高いものの、ラベルを返す。活性化関数は、ReLU 関数を用いている。出力層では、最大スコアのラベルを算出する。

### 3.3 攻撃評価環境

攻撃評価環境の概略を図 6 に示す。攻撃評価に用いた Mbed 用マイコンボードは、FRDM-K64F である。動作周波数は、最大 120MHz である。オシロスコープは、Tektronix DPO7104 を用いた。また、今回は、サイドチャネル情報として漏洩電磁波に着目した。電磁波プローブは、Langer RF2 U 5-2 を用いた。漏洩電磁波を計測している実験模様を図 7 に示す。

## 4. 評価実験

本章では、漏洩電磁波の計測により、学習モデルのアーキテクチャとパラメータを解析した結果を示す。攻撃の手順は、学習モデルのアーキテクチャを解析した後に、パラメータの解析を行う。

### 4.1 アーキテクチャ解析

アーキテクチャ解析では、層数、パーセプトロン数、活性化関数の種類について解析を行う。活性化関数の種類に関しては、机上検討とした。本実験では、学習モデルアーキテクチャの層構成の情報が、推論処理実行時の電磁波波形の形状からリークすることがわかった。

#### 4.1.1 層数

推論処理時に計測した電磁波波形から、SEMA(Simple Electromagnetic Analysis) で層数を解析する。計測した電磁波波形を図 8 に示す。2 層 MLP と 3 層 MLP の波形を示す。計測した電磁波波形は、1 回の動作で得られた単発の波形である。

図 8 が示す通り、電磁波波形の振幅の大きさとその形状から、スケールが異なる繰り返し処理の傾向が見える。2 層 MLP では、振幅の大きさ同じ部分が 3 つあり、その間の波形から、スケールが異なる繰り返し処理が観測できる。これは、2 層 MLP と 3 層 MLP に共通しており、3 層 MLP の方が、繰り返し処理の傾向が、1 つ多い。以上より、SEMA で学習モデルの層構成が判明しているといえる。

#### 4.1.2 パーセプトロン数

2 層 MLP の電磁波波形から、SEMA でパーセプトロン

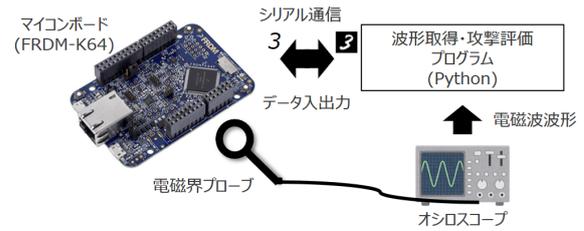


図 6 攻撃評価環境の概要

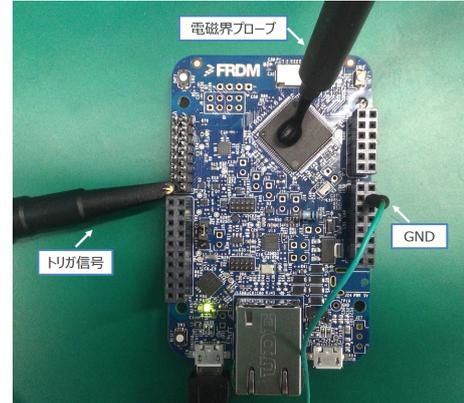


図 7 漏洩電磁波を計測している実験模様

数を解析する。はじめに、入力データが処理されるポイントを解析する。

エッジ AI デバイス向けには、省リソースを目的に、入力データや重みを量子化することが多い。Jacob らによると、float32bit から uint8 に入力データと重みを量子化しても、2%ほどの精度低下で済む [16]。一方で、バイアスについては、量子化することで、より精度が低下すると述べている。uTensor においても、入力データと重みを uint8 に量子化している。バイアスは、float32 のままである。Tan によると、uTensor における量子化では、CIFAR-10 データセット識別用学習モデルで、75%のメモリリソース削減、0.4%程度の精度低下に抑えている [17]。ここで、入力データ  $x$ 、量子化データ  $q$  における一般的な uint8 への量子化処理を式 (1) に示す。攻撃者は、入力データを入手できるため、式 (1) より、量子化データも計算可能である。

$$q = (x - \min(x)) * \frac{256}{\max(x) - \min(x)} \quad (1)$$

量子化データと電磁波波形との関係性から、量子化データを用いて演算するポイント、つまり、パーセプトロン処理のポイントを解析する。関係性の解析には、ANOVA F-test を用いた [18]。ANOVA F-test は、正規分布に従う複数群 (分散が等しいという仮定) で、平均が等しい (同じ母集団である) という検定である。量子化データと電磁波波形との関係性がある場合、量子化データで分類した電磁波波形グループの平均に差分があるといえる。F 統計量 (F value) は、K: グループ数、n: 各グループの要素数、N: 波形数 (サンプル数)、W: 波形データ、とすると式 (2) で

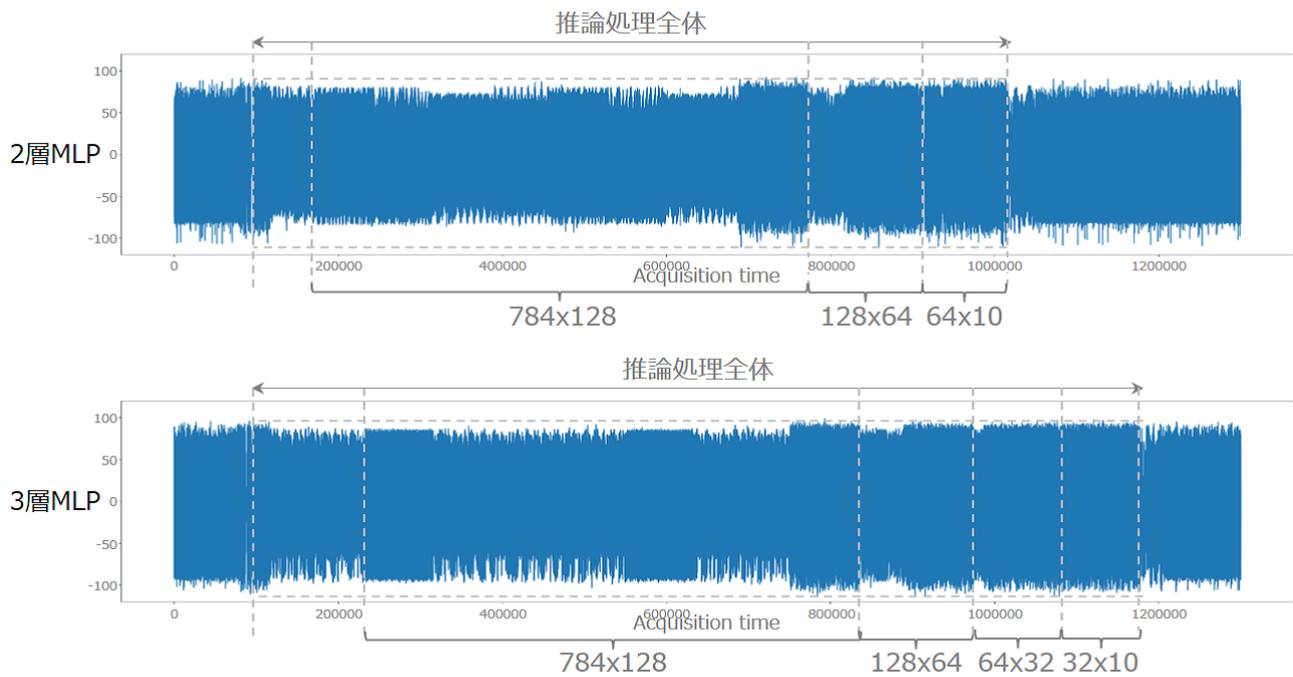


図 8 2層および3層 MLP における推論処理時の漏洩電磁波波形

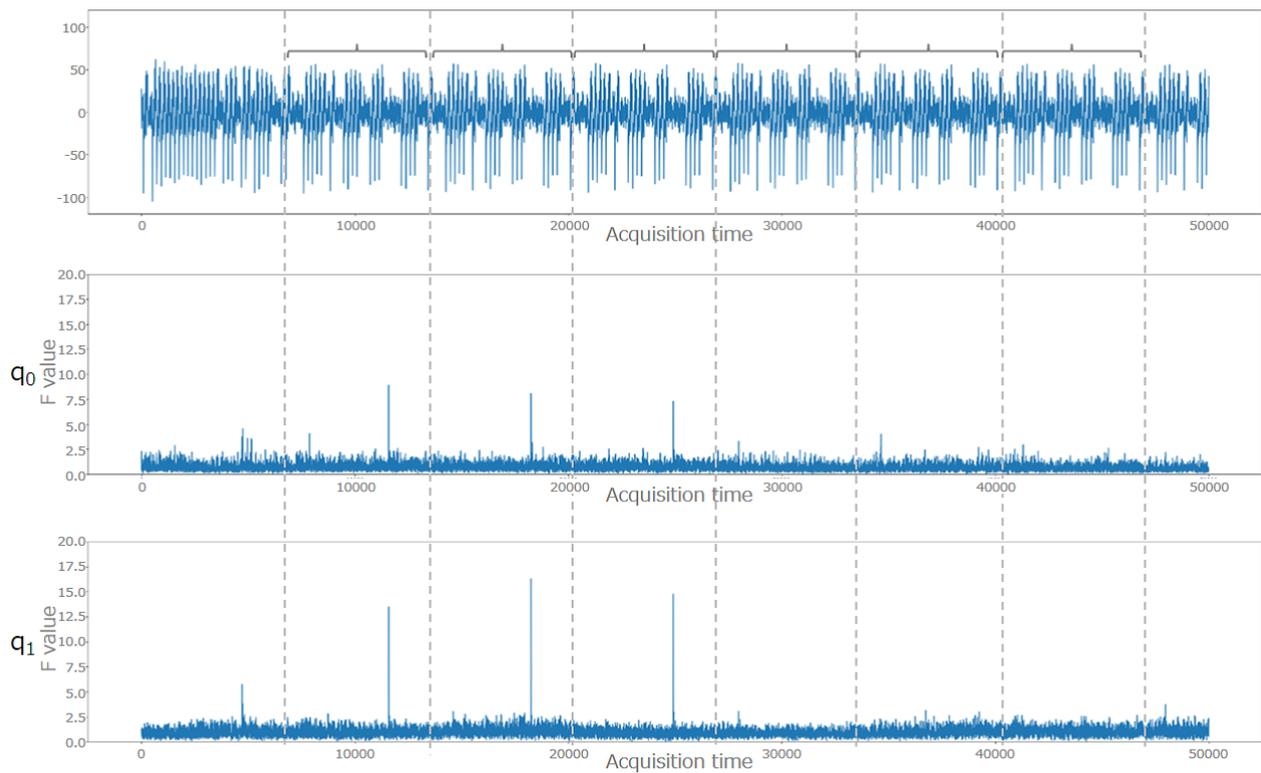


図 9 電磁波波形と入力データの量子化データ  $q_0, q_1$  における ANOVA F-test 結果  
(2層 MLP 全体波形から一部拡大 (200,000 ポイント付近))

算出できる.

$$Fvalue = \frac{SSA}{SSE} \quad (2)$$

$$SSA = \frac{1}{K-1} \sum_{i=1}^K n_i (\bar{W}_i - \bar{W}_N)^2 \quad (3)$$

$$SSE = \frac{1}{N-K} \sum_{i=1}^K \sum_{j=1}^{n_i} (W_{i,j} - \bar{W}_i)^2 \quad (4)$$

量子化データ  $q$  のハミング重みに着目して、電磁波波形における ANOVA F-test を行った結果を図 9 に示す。入力

データ  $x$  の  $x_0, x_1$  は、それぞれランダムな入力を 1,000 サンプル行い、その入力データに応じた電磁波波形 1,000 サンプルを取得した。図 9 は、取得した波形の平均波形と、入力データ  $x$  の  $x_0, x_1$  の量子化データ  $q_0, q_1$  における電磁波波形との ANOVA F-test の結果を示している。図 9 は、F 統計量が大きい部分 (各グループの平均に差分がある) を拡大している。電磁波波形も全体波形からの一部拡大である。図 9 より、 $q_0$  と  $q_1$  で F 統計量の最大値が入力の順番に出現していることがわかる。また、その時の電磁波波形に着目すると、波形の形状に繰り返し処理の傾向が見られる。したがって、図 9 から、ANOVA F-test より入力データに応じた処理のポイントと、電磁波波形から繰り返し演算の傾向が明らかになった。

入力データに関する繰り返し演算は、パーセプトロン処理における重みとの掛け算処理である。したがって、繰り返し演算の数をカウントすることにより、パーセプトロン数が算出できる。実験で用いた 2 層 MLP は、図 5 に示す通り、はじめに、784x128 回の乗算を行う。攻撃者は、入力データを把握しているため、入力データ数 784() と電磁波波形からカウントした乗算数 (784x128) から逆算することで、1 層目のパーセプトロン数 128 を明らかにできる。1 層目以降も、1 層目と同じ形状を示す電磁波波形を観測することで、同様にパーセプトロン数を解析できる。

#### 4.1.3 活性化関数の種類

電磁波波形における活性化関数の処理ポイントを解析するためには、その入力データを取得する必要がある。活性化関数の入力データは、図 3 の通り、重みとバイアスを知る必要がある。これらパラメータの解析は、次節で述べる。Batina らの攻撃手法 [3] によると、活性化関数の入力データを取得することで、入力データに応じた処理時間の傾向から活性化関数の種類を明らかにできる可能性がある。

### 4.2 パラメータ解析

パラメータ解析では、各パーセプトロンにおける重みとバイアスについて解析を行う。バイアスに関しては、机上検討とした。本実験では、重み情報のパラメータは、推論処理実行時の電磁波波形を相関係数を用いて統計処理することで情報を得ることがわかった。

#### 4.2.1 重み

4.1.2 節で特定した電磁波波形における乗算処理のポイントにおいて、CEMA (Correlation Electromagnetic Analysis) で重みを解析する。CEMA では、量子化データと重みの積と電磁波波形の相関を取る。相関係数 ( $\rho$ ) は、 $w$ : 推定重み、 $N$ : 波形数 (サンプル数)、 $W$ : 波形、 $H$ : 内部状態 (乗算結果のハミング重み) とする時、式 (5) で算出できる。

$$\rho(w) = \frac{\sum_{n=1}^N (W_n - \bar{W})(H_{n,w} - \bar{H}_w)}{\sqrt{\sum_{n=1}^N (W_n - \bar{W})^2} \sqrt{\sum_{n=1}^N (H_{n,w} - \bar{H}_w)^2}} \quad (5)$$

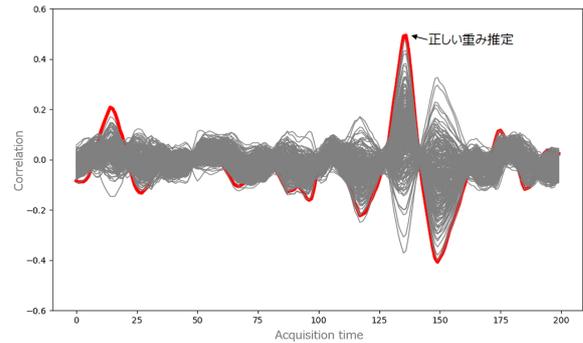


図 10 重み推定における相関係数の値

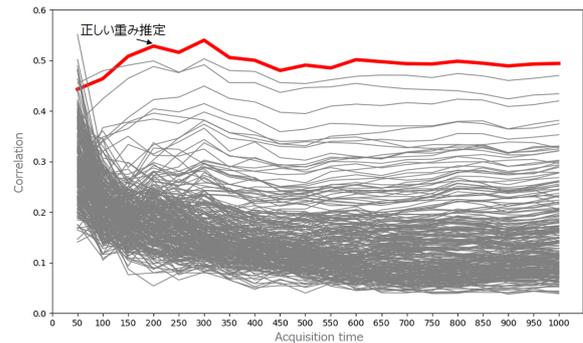


図 11 重み推定における波形数と相関係数の関係

電磁波波形における乗算処理のポイントにおいて、CEMA を行った結果を図 10 に示す。図 10 では、2 層 MLP における入力データ  $x_0$  の量子化データ  $q_0$  と推定した重みの積のハミング重みと電磁波波形 (1,000 サンプル) の相関を示している。重みは、量子化により、8bit である。したがって、図 10 は、256 個の重みの推定値における相関を示している。図 10 より、重みの正しい推定値で高い相関が得られている。

図 11 に CEMA による重みの解析に必要な波形数を示す。図 11 より、150 波形程度を用いれば、正しい重みを解析できることがわかる。攻撃者は、重み  $w_0$  以降の重みに関しても、入力データと乗算処理のポイントを変えることで解析できる。

#### 4.2.2 バイアス

電磁波波形におけるバイアスの処理ポイントを解析するためには、加算処理するデータを取得する必要がある。加算処理するデータは、図 3 の通り、複数の乗算結果を知る必要がある。また、バイアスに関しては、精度の観点で、量子化されることは少ないため、32bit の推定となる。32bit のパラメータの CMEA については、Batina らの攻撃手法 [3] によると、単精度浮動小数点数の形式から符号 + 指数部 9bit と仮数部 23bit に分けて、解析できる。ただし、Batina らは、32bit の重みを推定しており、バイアスについては言及していない。

## 5. 考察

評価実験から、アーキテクチャ解析とパラメータ解析について、攻撃者の事前知識の必要性の観点で、考察する。

### 5.1 アーキテクチャ解析

アーキテクチャ解析では、MLP 以外の CNN 等では、学習モデルに関して、攻撃者の事前知識が必要になる可能性がある。特に、畳み込み演算のパラメータや、枝刈りによる結線情報がサイドチャンネル情報から得られるかは、検討の余地がある。結線情報については、入力データに応じた重みとの乗算処理の順序から、解析できる可能性がある。例えば、入力データ  $x_0$  に関する処理の次が、 $x_2$  に関する処理となっている場合は、 $x_1$  に関する処理が枝刈りされていると推測される。

活性化関数の種類に関しては、事前の解析が必要である可能性がある。Batina らは、活性化関数の入力データに応じた処理時間の傾向から活性化関数の種類を明らかにした [3]。一方で、活性化関数の処理時間傾向は、別の実装でも同様な傾向を示すのかが疑問である。処理時間の傾向は、活性化関数の実装形態に大きく依存する。したがって、特定のデバイスや実装ごとに活性化関数の処理時間傾向をプロファイリングする必要があると言える。また、活性化関数の入力データの取得には、重みとバイアスを解析する必要があるため、解析の難易度は高い。一方で、活性化関数のパターンが限られているため、予測できる可能性もある。

### 5.2 パラメータ解析

パラメータ解析では、重み、バイアスともに、量子化処理の事前知識が必要になる可能性がある。組込み機器向けには、重み 8bit、バイアス 32bit の組み合わせが想定できる。

パラメータ解析は、比較的手間がかかる。表 2 に示す通り、2 層 MLP でも、109,184 個の重み、202 個のバイアスがある。今回の実験では、1 つの重みに対して、150 波形程度で解析できたが、その解析を重み数分、繰り返し行う必要がある。また、バイアスは、精度の観点で量子化されないため、解析に必要な波形数は、重みと比べて、増加すると考えられる。

## 6. おわりに

本論文では、実用的なニューラルネットワークを対象に、Model Reverse-Engineering 攻撃の脅威を評価した。MCU 向け深層学習開発プラットフォーム uTensor で開発された手書き文字識別用 DNN を対象に、漏洩電磁波を用いた Model Reverse-Engineering 攻撃を実施した。攻撃実施にあたり、攻撃者の前提知識がどれくらい必要か、また、攻撃の手順を整理した。実験結果により、実用的な MUC 実装

のニューラルネットワークにおいても、漏洩電磁波から層数や重みが漏洩しており、Model Reverse-Engineering 攻撃の脅威がある程度現実的であることを示した。

## 参考文献

- [1] Christian, S., Wojciech, Z., Ilya, S., Joan, B., Dumitru, E., Ian, G., and Rob, F.: *Intriguing properties of neural networks*, arXiv:1312.6199 (2013).
- [2] Florian, T., Fan, Z., Ari, J., Michael, R., and Thomas, R.: *Stealing machine learning models via prediction APIs*, USENIX Security Symposium (2016).
- [3] Lejla, B., Shivam, B., Dirmanto, J., and Stjepan, P.: *CSI Neural Network: Using Side-channels to Recover Your Artificial Neural Network Information*, arXiv:1810.09076 (2018).
- [4] uTensor, 入手先 (<https://github.com/uTensor>), (2019.08.22).
- [5] Nicholas, C. and David, W.: *Towards evaluating the robustness of neural networks*, arXiv:1608.04644 (2016).
- [6] Nicolas, P., Patrick, M., Ian, G., Somesh, J., Berkay, C., and Ananthram, S.: *Practical black-box attacks against machine learning*, ACM AsiaCCS (2017).
- [7] Weizhe, H., Zhiru, Z., and Edward, S.: *Engineering Convolutional Neural Networks Through Side-channel Information Leaks*, Annual Design Automation Conference (2018).
- [8] Mengjia, Y., Christopher, F., and Josep, T.: *Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures*, arXiv:1808.04761 (2018).
- [9] Sanghyun, H., Michael D., Yigitcan, K., Stuart, L., Ian, R., Kevin, K., Dana, D., and Tudor, D.: *Security Analysis of Deep Neural Networks Operating in the Presence of Cache Side-Channel Attacks*, arXiv:1810.03487 (2018).
- [10] Manaar, A. and Debdeep, M.: *How Secure are Deep Learning Algorithms from Side-Channel based Reverse Engineering*, arXiv:1811.05259 (2018).
- [11] Vasisht, D., Debasis, S., D Vijay, R., and Valentina, B.: *Stealing Neural Networks via Timing Side Channels*, arXiv:1812.11720 (2018).
- [12] Kota, Y., Takaya, K., Mitsuru, S., Takeshi, F.: *Model-Extraction Attack Against FPGA-DNN Accelerator Utilizing Correlation Electromagnetic Analysis*, IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (2019).
- [13] Yun, X., Zhuangzhi, C., Zuohui, C., Zebin, F., Haiyang, H., Jinyin, C., Yi, L., Zhefu, W., Qi, X., and Xiaoni, Y.: *Open DNN Box by Power Side-Channel Attack*, arXiv:1907.10406 (2019).
- [14] Mbed, 入手先 (<https://www.mbed.com>), (2019.08.22).
- [15] TensorFlow, 入手先 (<https://www.tensorflow.org>), (2019.08.22).
- [16] Benoit, J., Skirmantas, K., Bo, C., Menglong, Z., Matthew, T., Andrew, H., Hartwig, A., and Dmitry, K.: *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*, arXiv:1712.05877 (2017).
- [17] Neil, T.: *uTensor: How small can AI get?*, O'Reilly AI Conference (2018).
- [18] Shivam, B., Jean-Luc, D., Sylvain, G., and Zakaria, N.: *NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage*, International Symposium on Electromagnetic Compatibility (2014).