Vol.2021-CSEC-93 No.26
Vol.2021-IOT-53 No.26
2021/5/14

ReLUニューラルネットワークにおける Integrated GradientのVanilla Gradientへの帰着

三浦 尭之[1,a)]　権 英哲[2]　長谷川 聡[1]

概要：近年，深層学習をはじめとした機械学習の社会への浸透を背景に説明可能な AI に対する期待が高まっている．画像分類などでは，分類結果に対して重要だったピクセルをハイライトする勾配ベースの説明が盛んに研究されており，Sundararajan らによって提案された Integrated Gradient は Google Cloud などでサービスとして実装されている．一方で，最もシンプルな勾配ベースの説明である Vanilla Gradient は，訓練済みモデル保護の観点から脆弱性になりうるという報告がなされている．具体的には，入出力の情報から訓練済みモデルを盗み出す Model Extraction 攻撃が，出力に Vanilla Gradient による説明が加わることで効率的にできるという研究結果が報告されている．本研究では，活性化関数に ReLU が用いられているニューラルネットワークが Integrated Gradient による説明も出力する際，その情報から Vanilla Gradient の情報を復元できることを示した．これは，Integrated Gradeint による説明がついたモデルに対しても，説明が脆弱性になりうることを意味し，機械学習モデルのオープンな利活用を推進するにあたって検討しなければならない問題である．

1. はじめに

1.1 背景

近年，深層学習の発展や計算機の性能向上により，人工知能（Artificial Intelligence, AI）の処理能力が大きく向上した．その中でも，正規化線形関数（rectified linear function, ReLU [1]）は単純で計算量も小さく，それまでのスタンダードであったシグモイドなどと比べて学習が早く進む活性化関数で，深層学習の発展に大きく寄与し，現在，広く用いられている（VGG や ResNet など）．また，機械学習の開発インフラも整備が進んでおり，MLaaS（Machine Learning as a Service）などクラウド上で機械学習を行い，訓練済みモデルを展開する機会が増加している．

加えて，推論結果にその根拠となる情報も提示できる説明可能な AI（Explainable AI）にも注目が集まっており，診断補助や融資審査など幅広い分野での活用が期待されている [2]．特に深層学習が高い推論精度を発揮する画像分類では，勾配系の説明が主要な説明手法として用いられている（cf. 図 1）．勾配系の説明とは，モデルの入力に対する偏微分値などを用いて，「分類結果に対して重要だったピクセルをハイライトする」説明手法のことである．その中でも Integrated Gradient [3] という手法は Google Cloud のサービスでも主要な説明手法として用いられている [4]．

一方，機械学習を用いた AI にはそれ特有のセキュリティ脅威が提唱されており [5]，特に，訓練済み機械学習モデルを盗んでしまう Model Extraction 攻撃は，AI の社会展開を考える上で対策が求められる重要な課題である．Model Extraction 攻撃とは，訓練済み機械学習モデルに対して，その入出力の情報からそのモデルを復元する攻撃である．モデルが盗めることは，知的財産保護上の問題であり，またモデルをクラウド上展開しクエリ当たりの使用料金をとるビジネスにとって大きな脅威となる [6]．加えて，盗んだモデルを分析することで，モデルをだます Adversarial Examples を用いた攻撃や訓練データの情報を盗む Model Inversion 攻撃を効率的に行えるという報告もある [7]．

特に，近年期待が高まる説明可能な AI は，推論結果に加えて説明を出力する分だけ，Model Extraction 攻撃に脆弱な可能性があることが指摘されており，説明という利便性と盗みの脆弱性というトレードオフに注目が集まっている．実際，Vanilla Gradient という最もシンプルな勾配系の説明がついた深層学習モデルは，説明がないものと比べて Model Extraction 攻撃が効率的になってしまうことが報告されている [8], [9].

[1] NTT セキュアプラットフォーム研究所, 〒 180–8585 東京都武蔵野市緑町 3–9–11
[2] 東京大学大学院数理科学研究科
[a)] takayuki.miura.br@hco.ntt.co.jp

Vol.2021-CSEC-93 No.26
Vol.2021-IOT-53 No.26
2021/5/14

1.2 本稿の結果と貢献

本稿では，特に活性化関数が ReLU のニューラルネットワークにおいて，Integrated Gradient から Vanilla Gradient の情報を復元する手法を提案した．また，その手法を実装し実験することにより，実際に復元ができることを明らかにした．この手法により，Integrated Gradient つきのモデルに対しても，Vanilla Gradient つきのモデルと同様の Model Extraction 攻撃が行えることが明らかになった．Integrated Gradient つきのモデルは今後多くのサービスで用いられると考えられるため，本研究の結果より，AI の社会実装を考えてゆくうえで，より一層 Model Extraction 攻撃の対策が求められることが示された．

2. 準備

本稿の提案手法の記述・説明に必要な概念を紹介する．

2.1 深層学習

深層学習は，深いニューラルネットワーク（DNN）を確率的勾配降下法などを用いて学習させる機械学習の一手法である．

定義 2.1 (ニューラルネットワーク)．ニューラルネットワークとは，層状に並んだノード（ユニット）間を一方向に情報が伝わっていくような処理をするモデルであり，アフィン変換 $g_1, \ldots, g_l$ と活性化関数 $\sigma_1, \ldots, \sigma_l$ を用いて

$$f = \sigma_l \circ g_l \circ \cdots \circ \sigma_1 \circ g_1 : \mathbb{R}^d \to \mathbb{R}^c$$

と表せる．ここで，**アフィン変換** $g : \mathbb{R}^n \to \mathbb{R}^m$ とは，$m \times n$ 行列 A とベクトル $b \in \mathbb{R}^m$ で，

$$g(x) = Ax + b$$

と表せるような関数である．特にこの b を**バイアス項**という．

定義 2.2 (ReLU)．関数 $\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}$ を

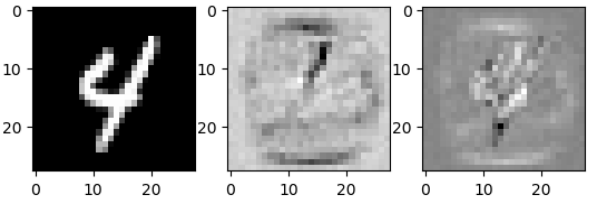
$$\mathrm{ReLU}(x) := \max(x, 0)$$

と定める．この関数を正規化線形ユニット（Rectified Linear Unit，以下 ReLU）と呼ぶ．また，入力がベクトル値の場合は，各成分にこの関数をかませた結果を並べたベクトルを出力するとする．活性化関数がすべて ReLU である DNN を本稿では ReLU **ニューラルネットワーク**と呼ぶ．

2.2 説明可能な AI

深層学習モデルは，画像分類などのタスクで高い推論精度を持つ一方で，特徴量抽出まで自動で行われるため，モデルの推論根拠が見えにくいという課題があった．こういった課題をうけて，そのモデルの推論根拠を人間にとって理解しやすくする説明可能な AI という技術に注目が集まっている．特に画像分類については，分類結果にとって重要だった画像のピクセルをハイライトする手法が主要な説明として用いられている（図 1）．

図 1　勾配系の説明



左から元データ，VG による説明画像，IG による説明画像
値が小さければ黒くなり，大きければ白くなるようになっている．

2.2.1 Vanilla Gradient

最もシンプルな勾配ベースの説明は次の Vanilla Gradient[10] である．

定義 2.3 (Vanilla Gradient)．機械学習モデル $f : \mathbb{R}^d \to \mathbb{R}$ がほとんど至るところ[*1]で微分可能とする．Vanilla Gradient では，$x \in \mathbb{R}^d$ に対して，

$$\mathrm{VG}_f(x) := \nabla f(x)$$

の値を説明として出力する．

2.2.2 Integrated Gradient

Vanilla Gradient は，与えられた入力 $x \in \mathbb{R}^d$ に対して，その点での勾配しか考えていない．そのため，分類結果に重要な影響を与えるピクセル x_i の貢献度がそこの近傍で“重要なまま変化しない”場合は $\frac{\partial f}{\partial x_i} = 0$（貢献度が低いことを意味する）となってしまう問題があった．Integrated Gradient [3] はこの問題をうけて，Sundararajanran らによって提案された説明手法であり，Google Cloud では実際にサービスとして実装されている [4]．

定義 2.4 (Integrated Gradeint(統合された勾配))．機械学習モデル $f : \mathbb{R}^d \to \mathbb{R}$ がほとんど至るところで微分可能とする．ベースラインと呼ばれる点 $x' \in \mathbb{R}^d$ を一つ固定する．Integrated Gradient では，$x \in \mathbb{R}^d$ に対して，

$$\mathrm{IG}_f(x, x') := (x - x') \odot \int_0^1 \nabla f(x' + t(x - x'))dt$$

の値を説明として出力する．ただし，ここで $\odot$ は各成分の要素ごとの積を表す記号とする．

特に，実装上はステップ数 m を定めて，

$$\mathrm{IG}^{approx}_{f,m}(x, x') = (x - x') \odot \frac{1}{m} \sum_{i=1}^{m} \nabla f(x' + \frac{i}{m}(x - x')) \tag{1}$$

という数値で代替している．

ただし，画像分類ではベースラインを $x' = 0 \in \mathbb{R}^d$ と選ぶことが多いため，本稿でも $x' = 0$ とすることとし，$\mathrm{IG}_f(x, x') = \mathrm{IG}_f(x)$ と略記する．

[*1] 全体集合に対してそうでないところのルベーグ測度（体積的なイメージ）が 0 であるという意味で使っている．

Vol.2021-CSEC-93 No.26
Vol.2021-IOT-53 No.26
2021/5/14

図 2　Model Extraction 攻撃の概略

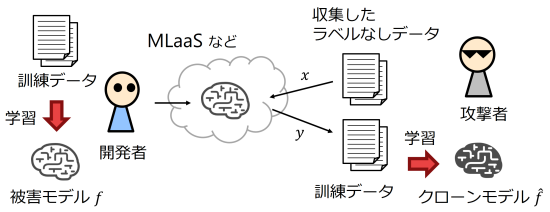
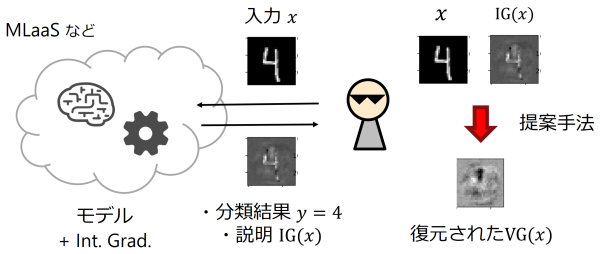


図 3　提案手法



2.3　Model Extraction 攻撃

訓練済み機械学習モデルに対して，その入出力の情報だけからそのモデルを復元してしまう攻撃を **Model Extraction 攻撃**と呼ぶ [6]．Model Extraction 攻撃は図 2 のような状況において，(1)〜(3) のような手順で行われる．ここで盗まれるモデルを被害モデル，攻撃者が復元したモデルをクローンモデルという．

(1) モデル開発者は独自に収集した訓練データを用いて，被害モデルを学習させ，クラウド上に展開する．
(2) 攻撃者は大量のラベルなしデータを収集し，被害モデルにクエリを投げラベルつきデータ（訓練データ）を作成する．
(3) 攻撃者は作成した訓練データで，自分の手元にクローンモデルを作成する[*2]．

2.3.1　説明可能な AI に対する Model Extraction 攻撃

盗みの対象となる被害モデルが説明可能な AI である場合，1 クエリ当たりの出力の情報が説明の分だけ多いため，盗みやすい可能性がある．実際，Milli ら [9] は Vanilla Gradient がついたモデルが盗みやすいことを実験で示し，三浦ら [8] は Vanilla Gradient の説明を生成モデルの学習に活かし，データフリーな Model Extraction 攻撃を効率的にできることを示した．また，勾配系の説明に限らず，Ulrich ら [11] は Counterfactual Explanation という事例ベースの説明を使って盗みの効率が上がることを示した．

著者らの知る範囲では Integrated Gradient による説明つきのモデルへの攻撃の先行研究は存在しないが，本稿の提案手法を応用すれば，Vanilla Gradient つきの ReLU ニューラルネットワークに対する攻撃を Integrated Gradient つきの ReLU ニューラルネットワークにも行うことができる．

3.　提案手法

本稿では，「Integrated Gradient による ReLU ニューラルネットワーク f の出力の説明 $\mathrm{IG}_f(x)$ から，Vanilla Gradient による説明 $\mathrm{VG}_f(x)$ を復元する手法」を提案する．

3.1　提案手法の概要

提案手法は図 3 のように用いることを想定している．具体的な手順は以下のとおりである；

(1) Integrated Gradient つきのモデルによるクラウド上のサービスに，利用者（攻撃者）はクエリとして $x \in \mathbb{R}^d$ を送る．
(2) サービスは推論結果 $f(x)$ と説明 $\mathrm{IG}_f(x)$ を返す．
(3) 利用者（攻撃者）は提案手法により，手元で $\mathrm{IG}_f(x)$ を $\mathrm{VG}_f(x)$ へと変換する．

3.2　ReLU ニューラルネットワークに関する既存結果

本項では提案手法の重要な前提となる「ReLU ニューラルネットワークの関数としての性質」を紹介する．具体例を可視化したものを付録 A.2 に記した．Rolnick らの研究 [12] から次の命題が成り立つ．

命題 3.1 (Rolnick)．$f : \mathbb{R}^d \to \mathbb{R}$ を ReLU ニューラルネットワークとする．このとき，有限個の超平面の和集合 H_f が存在し，$f|_{\mathbb{R}^d \setminus H_f}$ は各連結成分上で線形関数である[*3]．この連結成分 1 つ 1 つを f の（H_f に関する）**モデル線形領域**と呼ぶこととする．モデル線形領域は凸集合である．

系 3.2. 特に，f の各層のバイアス項が 0 であるならば，H_f を作る各超平面はすべて原点を通るようにできる．つまり各モデル線形領域の境界は原点を含むようにできる．

3.3　主定理

本稿の主定理を紹介する．関数 $f : \mathbb{R}^d \to \mathbb{R}$ を Integrated Gradient による説明がついた ReLU ニューラルネットワークとする．特に，バイアス項がない場合は f_{nobias} と書くこととする．

定理 3.3. 入力 $x \in \mathbb{R}^d$ が，任意のピクセルに対して $x_i \neq 0$ を満たすとする．このとき，ある $0 < \alpha < 1$ が取れて，αx が x と同じ f のモデル線形領域にあるとする．このとき，$\mathrm{IG}_f(x), \mathrm{IG}_f(\alpha x)$ とから $\mathrm{VG}_f(x)$ を復元することができる．

証明.　まず変数変換より次の式が成り立つ．

$$\begin{aligned}\mathrm{IG}_f(\alpha x) &= \alpha x \odot \int_0^1 \nabla f(t\alpha x)dt \\ &= x \odot \int_0^\alpha \nabla f(tx)dt\end{aligned}$$

[*2] ここで (2) と (3) は暫定版クローンモデルの様子を見て適応的にクエリを選ぶ場合もある．

[*3] この H_f の記述は本稿の議論に支障がない程度に簡略化してある．より正確な H_f の形を記述するにはいくつかの概念の整理が必要であり，詳しい情報は [12] を参照されたい．また，本稿の記述では H_f にあたる集合は一意に定まらないが，適宜一つ固定して議論を行うものとする

これより，次が成り立つ．

$$\mathrm{IG}_f(x) - \mathrm{IG}_f(\alpha x) = x \odot \int_\alpha^1 \nabla f(tx) dt$$

いま，x と αx を結ぶ線分は一つのモデル線形領域の中にあるので，その中では $\nabla f = \mathrm{VG}_f$ は定数．ゆえに，

$$\mathrm{IG}_f(x) - \mathrm{IG}_f(\alpha x) = (1-\alpha)x \odot \mathrm{VG}_f(x)$$

が成り立つ．いま，x の各成分は 0 ではないので，各成分の乗法的逆元を並べたベクトルを x^{-1} と置くと，

$$\mathrm{VG}_f(x) = \frac{1}{1-\alpha} x^{-1} \odot (\mathrm{IG}_f(x) - \mathrm{IG}_f(\alpha x))$$

というように復元できる．

□

これは，2 クエリがあれば，$\mathrm{IG}_f(x)$ を $\mathrm{VG}_f(x)$ に変換できることを意味するが，特にバイアス項がない f_{nobias} については次が成り立つ．

系 3.4. 入力 $x \in \mathbb{R}^d$ が，任意のピクセルに対して $x_i \neq 0$ を満たすならば，$\mathrm{IG}_{f_{\mathrm{nobias}}}(x)$ から $\mathrm{VG}_{f_{\mathrm{nobias}}}(x)$ を復元することができる．

証明. バイアス項がない場合は，系 3.2 より，任意の $\alpha > 0$ に対して，x と αx は同じモデル線形領域に入っている．そのため特に定理 3.3 の証明と同様に次の式が得られる；

$$\mathrm{VG}_{f_{\mathrm{nobias}}}(x) = x^{-1} \odot \mathrm{IG}_{f_{\mathrm{nobias}}}(x)$$

□

さらに，実装に注目すると次の定理を示せる．この定理は，ステップ数 m を指定して説明を受け取ることができるならば，2 クエリで IG の情報から VG の情報が復元できることを意味する．

定理 3.5. 入力 $x \in \mathbb{R}^d$ が, 任意のピクセルに対して $x_i \neq 0$ を満たすとする．Integrated Gradient による説明が, 式 *1* に基づいて実装されているならば, $\mathrm{IG}^{approx}_{f,m}(x), \mathrm{IG}^{approx}_{f,m-1}(\frac{m-1}{m}x)$ から，$\mathrm{VG}_f(x)$ を復元することができる．

証明. $\alpha = \frac{m-1}{m}$ とおく．定義より，次の 2 式が成り立つ

$$\mathrm{IG}^{approx}_{f,m}(x) = x \odot \frac{1}{m} \sum_{i=1}^{m} \nabla f(\frac{i}{m}x)$$

$$\begin{aligned}\mathrm{IG}^{approx}_{f,m-1}(\alpha x) &= \alpha x \odot \frac{1}{m-1} \sum_{i=1}^{m-1} \nabla f(\frac{i}{m-1}\alpha x) \\ &= x \odot \frac{1}{m} \sum_{i=1}^{m-1} \nabla f(\frac{i}{m}x)\end{aligned}$$

各辺を引くと，

$$\mathrm{IG}^{approx}_{f,m}(x) - \mathrm{IG}^{approx}_{f,m-1}(\alpha x) = x \odot \frac{1}{m} \nabla f(x)$$

が得られるので

$$\mathrm{VG}_f(x) = m x^{-1} \odot (\mathrm{IG}^{approx}_{f,m}(x) - \mathrm{IG}^{approx}_{f,m-1}(\alpha x))$$

が成り立つ．

□

この復元方法は，$m = \frac{1}{1-\alpha}$ から，定理 3.3 のインテグラルをシグマに変換しただけのように思えるが,「定理 3.5 は関数 f に対してほとんど仮定をしていない」という点が特筆すべき点である．

以上をまとめると，

- バイアス項がない場合

$$\mathrm{VG}_{f_{\mathrm{nobias}}}(x) = x^{-1} \odot \mathrm{IG}_{f_{\mathrm{nobias}}}(x) \tag{2}$$

- バイアス項がある場合
  - 理論的には

$$\mathrm{VG}_f(x) = \frac{1}{1-\alpha} x^{-1} \odot (\mathrm{IG}_f(x) - \mathrm{IG}_f(\alpha x)) \tag{3}$$

  - 実装的には

$$\mathrm{VG}_f(x) = m x^{-1} \odot (\mathrm{IG}^{approx}_{f,m}(x) - \mathrm{IG}^{approx}_{f,m-1}(\alpha x)) \tag{4}$$

という計算で復元できることがわかる．

4. 実験・評価

本節では，3 節にて提案した手法を実装し，計算機上でどの程度正確に Integrated Gradient の情報から Vanilla Gradient の情報が復元できるのかを確かめる．

4.1 目的

目的別に下記の 3 つの実験を行う；

- **実験 A**：系 3.4 の正当性の検証
  バイアス項なしのモデルに対して，式 2 で復元した $\mathrm{VG}_f(x)$ が真の $\mathrm{VG}_f(x)$ とどれくらい近いか評価する．
- **実験 B**：定理 3.5 の正当性の検証
  バイアス項があるモデルに対して，式 4 で復元した $\mathrm{VG}_f(x)$ が真の $\mathrm{VG}_f(x)$ とどれくらい近いか評価する．
- **実験 C**：定理 3.3 における α の値の観察
  バイアス項があるモデルに対して，式 3 で復元した $\mathrm{VG}_f(x)$ が真の $\mathrm{VG}_f(x)$ とどれくらい近いか評価する（実験 B において，二つのクエリのステップ数を $m = 50$ とそろえた場合）．

4.2 設定

実験の基本的な設定は以下のとおりである．実装は pytorch を用いて行った．

4.2.1 データセット

データセットには 0 ～ 9 の手書き文字のデータセット MNIST[13] を使用した．データの形式は 28×28 でグレースケールの画像で，$X = \{0, \ldots, 255\}^{28 \times 28}$ の元とみなせる．ここで黒が 0，白が 255 である．手書き文字のデータなので多くの入力画像はその輪郭側などの値が 0 である．しかし，データを読み込むとき正規化（0.1307 を引いて，0.3081 で割る）をしているので，入力データには $x_i = 0$ となるピクセルはないとみなせる．説明画像も入力データと同じ形式なので，$\mathbb{R}^{784}$ の元とみなせる．

4.2.2 モデル

実験に使用した ReLU ニューラルネットワークは 3 種類ある．いずれも関数としては $f : \mathbb{R}^{784} \to \mathbb{R}^{10}$ である．本稿では $f : \mathbb{R}^d \to \mathbb{R}$ のモデルを理論的に考察しているため，入力の正解ラベルにあたる成分の出力のみに注目して，$f : \mathbb{R}^{784} \to \mathbb{R}$ とみなすこととする[*4]．各モデルの構造や各層の説明は付録 A.1 に記した．

- `TwoLayerNet`（表 A·1）
  2 層の全結合層からなるモデル．バイアス項なしのモデル $\texttt{TwoLayerNet}_{\text{nobias}}$ はテスト精度 95.77%，バイアス項あり `TwoLayerNet` は 95.69%である．
- `LeNet`（表 A·2）[14]
  前半 2 層が畳み込み層で後半 2 層が畳み込み層になっているモデル．バイアス項なしのモデル $\texttt{LeNet}_{\text{nobias}}$ はテスト精度 98.81%，バイアス項あり `LeNet` は 98.92%である．
- `ResNet18`（表 A·3）[15]
  最初に 1 層の畳み込み層，Basic-Block という 2 層の畳み込み層からなる塊を 8 個，最後に全結合層をつなげた 18 層のモデル．途中に入るバッチノーマライゼーションの層 [16] は，フォワード計算の際はバイアス項ありアフィン変換に相当するので，ResNet はバイアス項ありのみを考えることとする．テスト精度は 99.42%である．

4.2.3 説明手法の実装

Vanilla Gradient VG_f は pytorch の autograd 機能を用いて実装した．また，Integrated Gradient $\mathrm{IG}^{approx}_{f,m}$ は autograd と式 1 により実装した．近似のステップ数は特に断りがない場合は $m = 50$ とする．

4.2.4 評価方法

MNIST のテスト用データ 10000 件それぞれに対して，真の $\mathrm{VG}_f(x)$ と，$\mathrm{IG}^{approx}_{f,m}(x)$ から復元した Vanilla Gradient $\mathrm{VG}_f(x)$ を，それぞれベクトルとして長さを 1 に正規化して二乗距離を測り，その平均値で評価する．距離は 0～2 の間をとりうるということがわかり，また，2 つのベクトルがランダムにとられた場合の距離の期待値は $\sqrt{2} \approx 1.414$ であることもわかる．

[*4] この考察をするために，実験では最後の出力の層の softmax 関数を省略している．sotfmax をかませる出力間の相互作用が起きてしまうからである．また，そのため用いたモデルは通常より多少テスト精度が落ちている．

表 1 各モデルの正規化した $\mathrm{VG}_f(x)$ と $\mathrm{IG}_f(x)$ の平均距離

モデル	平均誤差	誤差の分散
$\texttt{TwoLayerNet}_{\text{nobias}}$	1.682	1.427×10^{-1}
`TwoLayerNet`	1.618	1.468×10^{-1}
$\texttt{LeNet}_{\text{nobias}}$	1.454	9.211×10^{-2}
`LeNet`	1.339	8.050×10^{-2}
`ResNet18`	1.238	9.749×10^{-2}

表 2 実験 A 結果

モデル	平均誤差	誤差分散
$\texttt{TwoLayerNet}_{\text{nobias}}$	3.708×10^{-7}	2.058×10^{-7}
$\texttt{LeNet}_{\text{nobias}}$	2.028×10^{-3}	1.245×10^{-2}

表 3 実験 B 結果

モデル	平均誤差	誤差分散
`TwoLayerNet`	2.028×10^{-4}	1.414×10^{-2}
`LeNet`	1.175×10^{-2}	3.504×10^{-2}
`ResNet`	1.198	4.926×10^{-1}

表 4 実験 C 結果

モデル	平均誤差	誤差分散
`TwoLayerNet`	2.478×10^{-2}	4.666×10^{-2}
`LeNet`	8.368×10^{-1}	9.391×10^{-2}
`ResNet`	1.392	2.587×10^{-1}

4.3 結果

4.3.1 比較用

各実験で得る二乗距離がどの程度のものなのか比較するために，各モデルに対して，$\mathrm{VG}_f(x)$ と，$\mathrm{IG}^{approx}_{f,m}(x)$ をそれぞれ正規化したものの二乗距離の平均（平均誤差）をとった．結果は表 1 である．

4.3.2 実験結果

実験結果は表 2, 3, 4 のとおりである．ここで実験 B ではステップ数は入力画像を $m = 50$ として式 4 の通りに行った．つまり摂動させた画像のクエリは $m - 1 = 49$ として行った．実験 C については，$m = 50$，$\alpha = 0.98$ としたときの結果である．また，各実験での画像の例は表 4 に記した．

4.4 考察

4.4.1 `TwoLayerNet`, `LeNet` について

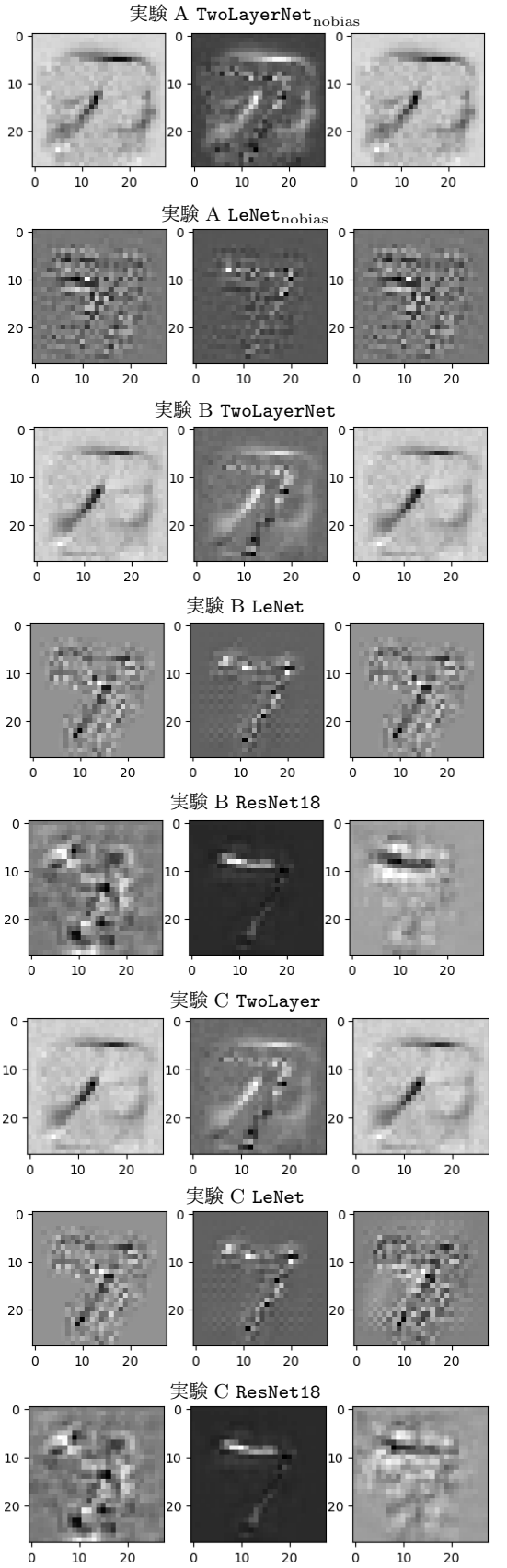
実験 A，B，C を見るとニューラルネットワークが 2 層，4 層の場合は，どちらも誤差が小さく，理論通り正確に復元できているといえる．実際，図 4 を見てもかなり正確に説明画像が復元できていることが眼でも確認できる．

4.4.2 `RenNet18` について

実験 B，C において，`RenNet18` があまり正確に復元できなかったという結果の原因として，桁落ちが考えられる．Integrated Gradient では，ステップ数 m を定め，入

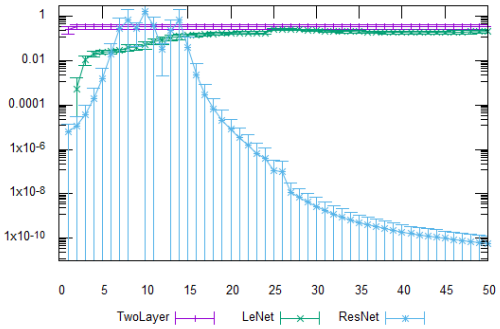
Vol.2021-CSEC-93 No.26
Vol.2021-IOT-53 No.26
2021/5/14

図 4　実験 A，B，C の一例



左から $\mathrm{VG}_f(x)$，$\mathrm{IG}_f(x)$，$\mathrm{IG}_f(x)$ から復元された $\mathrm{VG}_f(x)$

図 5　$||\nabla f(x)||$ の値



力画像 x に対して，$y = \frac{1}{m}x, \frac{2}{m}x, \frac{3}{m}x, \ldots, x$ の各点での勾配 $\nabla f(y)$ を計算し，$\frac{1}{m} \odot x \nabla f(y)$ を足し合わせている．そのうち，特に今回取り出したいのは $y = x$ の場合の勾配 $\nabla f(x)$ であるが，`ResNet18` に関しては，$\nabla f(x)$ がそれ以前（特に前半の）$\nabla f(\frac{i}{m}x)$ と比べて小さいことが図 5 からわかる．縦軸を対数目盛に取っているが，特に $i = 10$ 前後に対して，$i = 50$ の時はその 10^{-10} 倍になっていることが確認できる．

この現象を回避するためには計算機上で使える有効桁数を大きくする必要がある．しかし，モデル推論の説明には，そこまで詳細な桁数は必要ないと考えられるため，本稿の提案手法の単純な適用では ResNet18 に対しては，そこまで正確に情報が復元できないことを意味する．

この現象とモデルの層が深いことの関係は今後考察が求められる．モデルの深さ以外にも入力 x の種類と勾配 $\nabla f(x)$ の大きさの関係なども同様に考察が求められる．

5. まとめ

本研究では，ReLU ニューラルネットワークにおいて，Integrated Gradient による説明の情報から Vanilla Gradient による説明の情報を復元する手法を提案した．この事実により，Integrated Gradient による説明がついたモデルにも，Vanilla Gradient による説明がついたモデルと同様の Model Extraction 攻撃の脅威があることが明らかになった．これは多くの実装されている深層学習モデルが ReLU ニューラルネットワークであり，Integrated Gradient の実際のサービスも展開されていることから，説明可能な AI に対する Model Extraction 攻撃の脅威がより大きいものであることを表し，その早急な対策がもとめられる．

また，本手法に関する技術的な残された課題としては，「softmax がある場合のモデルに対しての適用手法の考案」，「深い層のモデルと勾配の関係の観察」「モデル固定下での入力と勾配の関係の観察」などがあげられる．

参考文献

[1] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[2] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, Vol. 6, pp. 52138–52160, 2018.
[3] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.
[4] Google Cloud, 「AI Platform の AI Explanations の概要」. `https://cloud.google.com/ai-platform/prediction/docs/ai-explanations/overview?hl=ja`（最終閲覧日：2021 年 3 月 29 日）.
[5] Mingfu Xue, Chengxiang Yuan, Heyi Wu, Yushu Zhang, and Weiqiang Liu. Machine learning security: Threats, countermeasures, and evaluations. *IEEE Access*, Vol. 8, pp. 74720–74742, 2020.
[6] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 601–618, 2016.
[7] Yi Shi, Yalin E Sagduyu, Kemal Davaslioglu, and Jason H Li. Active deep learning attacks under strict rate limitations for online api calls. In *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–6. IEEE, 2018.
[8] 三浦尭之, 長谷川聡. 説明可能な AI に対するデータ収集を必要としない model stealing 攻撃. *Symposium on Cryptography and Information Security*, 2021.
[9] Smitha Milli, Ludwig Schmidt, Anca D Dragan, and Moritz Hardt. Model reconstruction from model explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 1–9, 2019.
[10] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
[11] Ulrich Aïvodji, Alexandre Bolot, and Sébastien Gambs. Model extraction from counterfactual explanations. *arXiv preprint arXiv:2009.01884*, 2020.
[12] David Rolnick and Konrad Kording. Reverse-engineering deep relu networks. In *International Conference on Machine Learning*, pp. 8178–8187. PMLR, 2020.
[13] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.
[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

付　　録

A.1 実験モデルの構造

本稿の実験で用いたニューラルネットワークのモデル構造の説明をする．基本的なアフィン変換はすべてのノードがつながっていると考えて「全結合層（fc)」と呼ばれる．

DNN は基本的には，全結合層と活性化関数の繰り返しであるが，実際のモデルでは他の処理も行われる．本稿で用いた `TwoLayerNet`，`LeNet`，`ResNet` の中に入っている層を紹介する．各層入力データの形式を $n = W_{\text{in}} \times W_{\text{in}} \times C_{\text{in}}$，出力データの形式を $m = W_{\text{out}} \times W_{\text{out}} \times C_{\text{out}}$ とする．

定義 A.1.1 (畳み込み層, conv)．カーネルサイズ $\text{Ker} \in \mathbb{N}$，ストライド $s \in \mathbb{N}$ の畳み込み層 $\text{conv} : \mathbb{R}^n \to \mathbb{R}^m$ とは，$y = \text{conv}(x)$ に対して，各 $0 \leq i, j \leq W_{\text{out}} - 1, 0 \leq c \leq C_{\text{in}} - 1, 0 \leq m \leq C_{\text{out}} - 1$ が，

$$y_{ijm} = \sum_{p,q=0}^{\text{Ker}-1} x_{si+p,sj+q,c} h_{pqcm}$$

となる処理を行う層である．これはアフィン変換とみなせる．ここで，モデルのパラメータは $\{h_{pqcm}\}$ である．

また，本稿では入出力の形式を適宜合わせるため，余白を拡張する *padding* は適宜行われているものとする．

定義 A.1.2 (最大プーリング層, pool)．カーネルサイズ $\text{Ker} \in \mathbb{N}$，ストライド $s \in \mathbb{N}$ とする．ここで，**最大プーリング層** $\text{pool} : \mathbb{R}^n \to \mathbb{R}^m$ とは，$y = \text{pool}(x)$ に対して，各 $0 \leq i, j \leq W_{\text{out}} - 1, 0 \leq c \leq C_{\text{in}} - 1, 0 \leq m \leq C_{\text{out}} - 1$ が，

$$y_{ijm} = \max_{0 \leq p,q \leq \text{Ker}-1} x_{si+p,sj+q,c}$$

となる処理を行う層である．これは広義の ReLU のように解釈できるため，本稿の考察には支障がない．

定義 A.1.3 (バッチノーマライゼーション層，bn)．深層学習では，入力データ一つ一つに対して勾配を計算していくわけではなく，64 個や 128 個などを一塊とし（これをバッチと呼ぶ）これを並列で各層の処理を行う．バッチノーマライゼーション層ではバッチ間のデータの平均 μ，分散 σ を計算し，各データを $x \to (x - \mu)/\sigma$ と変換する．学習時は μ と σ をパラメータとみなし学習によって更新していくが，推論時は固定されるためアフィン変換とみなせる．

定義 A.1.4 (ショートカットコネクション)．ResNet は Basic-Block と呼ばれる二回畳み込みを行う層 $BB : \mathbb{R}^n \to \mathbb{R}^m$ を複数つなげたモデルである．この Basic-Block では畳み込みとバッチノーマライゼーションを行うが，その出力に対して，元データ（を適宜そのまま変形したもの）を加えたものを用いて

$$y = BB(x) + x$$

という処理を行う．この仕組みを**ショートカットコネクション**と呼ぶ．

`TwoLayerNet` は 2 層，`LeNet` は 4 層，`ResNet` は 18 層の DNN であるが，ここで層としてカウントされているのは，全結合層 fc, 畳み込み層 conv である．

表 A·1 TwoLayerNet

名称	stride	kernel	出力形式	活性化関数
data	–	–	$28 \times 28 \times 1$	–
fc1	–	–	$100 \times 1 \times 1$	ReLU
fc2	–	–	$10 \times 1 \times 1$	–

表 A·2 LeNet

名称	stride	kernel	出力形式	活性化関数
data	–	–	$28 \times 28 \times 1$	–
conv1	1	3×3	$26 \times 26 \times 32$	ReLU
conv2	1	3×3	$24 \times 24 \times 64$	ReLU
pool1	2	2×2	$12 \times 12 \times 64$	–
fc1	–	–	$1 \times 1 \times 128$	ReLU
fc2	–	–	$1 \times 1 \times 10$	–

表 A·3 ResNet18

名称	stride	kernel	出力形式	活性化関数
data	–	–	$28 \times 28 \times 1$	–
conv1	2	7×7	$14 \times 14 \times 64$	
bn1	–	–	–	ReLU
pool1	2	3×3	$7 \times 7 \times 64$	–
bb1	$s = 1$	–	$7 \times 7 \times 64$	–
bb2	$s = 1$	–	$7 \times 7 \times 64$	–
bb3	$s = 2$	–	$4 \times 4 \times 128$	–
bb4	$s = 1$	–	$4 \times 4 \times 128$	–
bb5	$s = 2$	–	$2 \times 2 \times 256$	–
bb6	$s = 1$	–	$2 \times 2 \times 256$	–
bb7	$s = 2$	–	$1 \times 1 \times 512$	–
bb8	$s = 1$	–	$1 \times 1 \times 512$	–
fc	–	–	$1 \times 1 \times 10$	–

表 A·4 Basic-Block (s, c_{out})

名称	stride	kernel	出力形式	活性化関数
data	–	–	$w \times w \times c_{\text{in}}$	–
conv1	s	3×3	$\lceil \frac{w}{s} \rceil \times \lceil \frac{w}{s} \rceil \times c_{\text{out}}$	–
bn1	–	–	–	ReLU
conv2	1	3×3	$\lceil \frac{w}{s} \rceil \times \lceil \frac{w}{s} \rceil \times c_{\text{out}}$	–
bn1	–	–	–	ReLU

ここで，$\lceil x \rceil$ は x 以上の最小の整数．

A.2 ReLU ニューラルネットワークの幾何学的解釈

2 層でバイアス項なしの ReLU ニューラルネットワークを例に既存定理 3.1，3.2 のイメージを紹介する．2 層の場合ニューラルネットワークは $f : \mathbb{R}^d \to \mathbb{R}^h \to \mathbb{R}$ で特に，

$$f(x) = \sum_{i=1}^{h} w_i \max({}^t A_i x, 0) \qquad (A.1)$$

と表せる．ここで，$A_1, \ldots, A_h \in \mathbb{R}^d, w \in \mathbb{R}^h$ である．次のような特性関数 $g : \mathbb{R}^d \to \{0,1\}^h$ を用意する；

$$g_A(x)_i = \begin{cases} 1 & ({}^t A_i x > 0) \\ 0 & (otherwise). \end{cases}$$

すると式 A.1 は

$$f(x) = \sum_{i=1}^{h} w_i g_A(x)_i {}^t A_i x$$

と書き直せるが，ここで，各 $s \in \{0,1\}^h$ に対して

$$A_s := \sum_{i=1}^{h} w_i s_i A_i \in \mathbb{R}^h$$

とおく．すると，2 層の ReLU ニューラルネットワークは

$$f(x) = {}^t A_{g_A(x)} x$$

と表せる．

各 A_i について，直交補空間は $A_i^{\perp} := \{x \in \mathbb{R}^d \mid {}^t A_i x = 0\}$ であり，原点を通る超平面になっている．$A_1, \ldots, A_h$ が一次独立ならば，この h 個の超平面によって入力空間 $\mathbb{R}^d$ は 2^h 個の部屋に分割され，その部屋ごとに A_s が対応している．そのため，ReLU ニューラルネットワークはこの部屋の中ではただのベクトル A_s による線形写像になっていることがわかり，この部屋がモデル線形領域である．3 層以上のモデルでは，この超平面が所々で折り曲げられている形になっており，バイアス項がある場合はこの折れ曲がった超平面は原点を通らない．

例 A.2.1. $f : \mathbb{R}^3 \to \mathbb{R}^2 \to \mathbb{R}$ で考える．すると，

$$f(x) = \begin{cases} {}^t A_{(1,1)} x = {}^t(w_1 A_1 + w_2 A_2) x & (g_A(x) = (1,1)) \\ {}^t A_{(1,0)} x = {}^t(w_1 A_1) x & (g_A(x) = (1,0)) \\ {}^t A_{(0,1)} x = {}^t(w_2 A_2) x & (g_A(x) = (0,1)) \\ {}^t A_{(0,0)} x = 0 & (g_A(x) = (0,0)). \end{cases}$$

と表現できる．

図 A·1 例 A.2.1 の入力空間の幾何的なイメージ