「マルチメディア，分散，協調とモバイル
(DICOMO2019) シンポジウム」 令和元年7月

再帰型ニューラルネットワークへのモデル抽出攻撃の精度評価

竹村 達也[1,a)] 矢内 直人[1,b)] 藤原 融[1,c)]

概要：モデル抽出攻撃は, 公開的にアクセス可能な学習モデルに対し, そのクエリアクセスを通じて攻撃者がより少ない計算資源およびデータ量で学習モデルと同等以上の性能を持つモデルを得る攻撃である. 既存のモデル抽出攻撃に対する研究は 3 層の深層ニューラルネットワークなど単純なモデルに対してのみしか行われておらず, 音声認識など時系列データを扱う再帰型ニューラルネットワーク (RNN) に関してどのような脅威が起こりえるか自明ではない. 本稿では RNN へのモデル抽出攻撃の脅威の把握として, 複雑かつ高性能な RNN である長・短記憶 (LSTM) に対し, 単純な RNN を用いて, より高い精度のモデルが抽出できるか明らかにする. 具体的には, 二つの問題設定において議論する. まず, 画像認識などに代表される分類問題の設定において, LSTM の中間出力を用いることで, 最終出力を待たずにモデルが抽出できることを示す. 次に, 時系列データを主に扱う回帰問題の設定において, 新たな損失関数の設計を通じた攻撃方法を提案する. MNIST データセットおよび Air Quality データセットを用いて実験したところ, MNIST では データセット総数の 20% の訓練データ数で精度 97.5 % のモデルが, Air Quality では データセット総数の約 60 % の訓練データ数で精度 88.3 % のモデルがそれぞれ抽出できることを確認した.

キーワード：深層学習, モデル抽出攻撃, 再帰型ニューラルネットワーク, 時系列データ

Accuracy evaluation of Model Extraction Attacks against Recurrent Neural Networks

Tatsuya Takemura[1,a)] Naoto Yanai[1,b)] Toru Fujiwara[1,c)]

1. 序論

1.1 背景

深層学習 [1] は様々な分野において高い効果を挙げている, 機械学習の最先端領域である. 深層学習はとくに計算負荷が大きいことから, Machine-Learning-as-a-Service (MLaaS) と呼ばれる, クラウドサーバを介してモデルの機能を提供するビジネス形態が近年の主流になりつつある[*1][*2]. ここでいうモデルとは学習と予測の二つの処理からなるものであり, サービスの提供者は自らが学習したモデルをクラウドに保管し, クライアントはクラウドへの API を介してモデルに予測処理を依頼する.

このとき, API を介した予測処理の実施は, モデルに関する何らかの情報を漏らす可能性がある. このような観点において近年高い関心を受けている攻撃がモデル抽出攻撃 [2], [3] である. これはクライアントに扮した攻撃者がモデルへの API アクセスおよびその予測結果を通じて, 手元に保存している自分のモデルを学習させる攻撃である. このとき, 攻撃者の利点としてはサービスの提供者よりも保有するデータ量が少ないにもかかわらず, より高い精度のモデルを得ることが可能となる [4], [5]. 一般には学習にかかるデータの収集および学習処理そのものは負荷の大きい作業であることから, 攻撃者の経済的利益は大きいものとなる. また, 文献 [5] によると, モデル抽出攻撃の応用として, この抽出したモデルを通じてどのような予測なら誤りやすいか分析できるような transferrable adversarial example [6] も検討されている. これらの観点は機械学習お

[1] 大阪大学 大学院情報科学研究科
[a)] t-tatsuya@ist.osaka-u.ac.jp
[b)] yanai@ist.osaka-u.ac.jp
[c)] fujiwara@ist.osaka-u.ac.jp
[*1] https://aws.amazon.com/jp/aml/
[*2] https://azure.microsoft.com/ja-jp/services/machine-learning-studio/

よび深層学習に対する信頼を根幹から揺るがすような極めて重要な問題といえる.

しかしながら, このモデル抽出攻撃の重要性に関わらず, 既存研究では線形回帰問題 [2] や 3 層の深層ニューラルネットワーク (DNN) [2], [5], [7] など簡単なアーキテクチャでしか検討されていない. これはアーキテクチャごとの違いが分かっておらず, たとえば再帰型ニューラルネットワークなど言語処理で使われるような複雑なモデルでは, どの程度の脅威があるのか非自明であることを意味する. とくに深層学習ではアーキテクチャごとに計算処理が大きく異なるため, この違いは攻撃者における抽出攻撃の成功条件や得られる効果に影響する可能性も予想される. 上述したとおり, 近年では adversarial example の分析なども応用としてありえることから, 様々なアーキテクチャに対し抽出攻撃の検討を行っていくことは極めて重要な研究課題といえる.

1.2 貢献

本稿では再起型ニューラルネットワーク (RNN) および長・短期記憶 (LSTM) といったニューラルネットワークに対し, 画像分類などに代表される分類問題および株価予測などに代表される回帰問題それぞれの設定でモデル抽出攻撃を行う. また, とくに分類問題の設定において, 攻撃者が攻撃対象のモデルより高い精度を持つモデルを得られることを示す. 本稿の技術的貢献は二つである. まず第一の貢献は分類問題の設定において, 再帰型ニューラルネットワークの特性を活用することで, 攻撃対象のモデルが最終出力を計算するよりも速く, 攻撃者が自分のモデルを学習できる可能性を示した点である. 次に, 回帰問題の設定では分類問題を対象とした既存研究の手法 [4] が使えないことから, 回帰問題用の攻撃によらせた損失関数の構成を新たに示している. これが二点目の貢献である.

以下に, 本稿の攻撃の直観を述べる. まず RNN および LSTM では時刻ごとに中間値となる出力を求め, その結果を以降の時刻における計算にフィードバックすることで最終出力を求めていく. これらの中間値および最終出力に対し偏りを持たせた学習を行うことで, 攻撃者観点では少ないデータ量における高い精度での学習が期待できる. 分類問題においては, とくに中間値を利用することで, 攻撃者は LSTM からの最終出力を待たずに, より高い精度で自らの RNN の学習が可能となる. 一方, 回帰問題においては, LSTM からの中間値が攻撃者の与える入力に相当することから, 中間値の情報が RNN 側の学習に関する利点を得ることができない. このため, LSTM からの最終出力に対してのみ偏りを持たせることで, RNN の学習に関わるデータ量の削減を図る. 詳細については 3.2 節を参照されたい. 実際に MNIST データセットおよび Air Quality データセットを用いて実験したところ, MNIST データセットではデータセット総数の 20% の訓練データ数で精度 97.5 % のモデルが, Air Quality データセットでは 6 ヶ月分の訓練データ数で精度 88.3 % のモデルがそれぞれ抽出できることを確認した.

1.3 関連研究

モデル抽出攻撃は文献 [2] で初めて報告された. ここではロジスティック回帰に対してモデル抽出攻撃を行っていた. 岡田ら [4] は画像分類を行う DNN および畳み込みニューラルネットワーク (CNN) に対してモデル抽出攻撃を行っており, 温度付きソフトマックス関数 [9] を用いた攻撃手法を示すことで, 攻撃対象のモデル以上の精度を得ることに成功している. 著者らの知る限り, CNN などのような複雑なニューラルネットワークを扱っている点, また, 温度付きソフトマックス関数を用いてより高い精度のモデルを得ている点において, 岡田らの成果が最新の研究といえる. このため, 岡田らの成果と主に比較する.

モデル抽出攻撃への対策として, 文献 [8] では抽出警告を出す研究がなされている. クラウド上で攻撃者が学習しているであろうモデルをプロキシを用いて疑似的に学習し, API の前後で情報利得を計算し, 閾値を超えると警告するというものである. しかしながら閾値について議論はされていない. 文献 [5] において詳細に DNN への警告について議論されているが, 著者らの知る限り, RNN では議論されていない.

モデル抽出攻撃に類似した手法として, ニューラルネットワークの蒸留 [9], [10], [11], モデルの圧縮 [12] が知られている. これは大規模または複数のニューラルネットワークモデル (教師モデル) が学習した内容を, 計算コストを削減するために小規模なニューラルネットワーク (生徒モデル) に圧縮するものである. 圧縮・蒸留とモデル抽出攻撃の違いは攻撃者におけるデータ量である. 圧縮や蒸留において生徒モデルは必要な訓練データ量と精度を上げるために教師モデルから情報を存分に取ることができるが, モデル抽出攻撃の場合は, できるだけ少ない訓練データ量, かつ, できるだけ少ない教師モデルへのアクセスで行う必要がある.

さらなる関連研究として, 文献 [13] では Intel SGX などのようなハードウェアセキュリティメカニズムで局所的に分離されたモデルを攻撃対象として, ハードウェアサイドチャネル攻撃を導入している. 文献 [13] の著者らによると, この攻撃を踏み台にすることでモデル抽出攻撃を効果的に行うことが予想されている.

2. 準備

2.1 ニューラルネットワークにおいて扱う問題

2.1.1 分類問題

入力に対して, 出力がどの部類に分類されるかを予測す

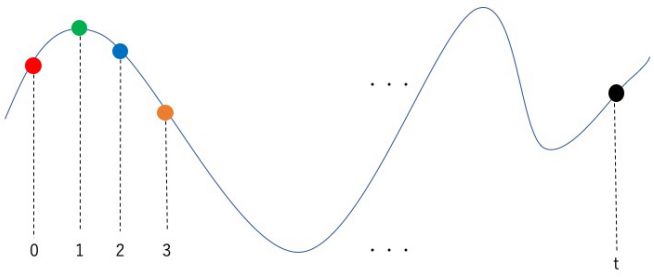

図 1　音声の波形の例

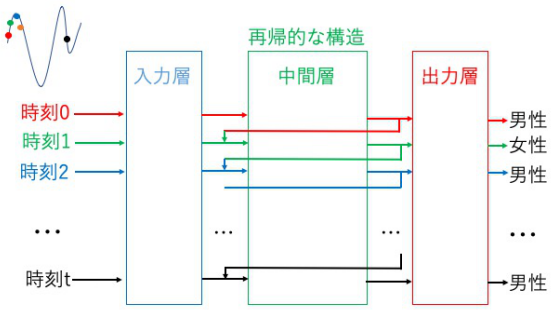

図 2　再帰型ニューラルネットワークのアーキテクチャ

る問題である. 例えば, 手書き数字の認識では, 入力として与えられた手書き数字のデータが 0 から 9 のうちのどれかを推測するニューラルネットワークによって解くことができる. 分類問題を解くニューラルネットワークは出力層に分類結果の数だけニューロンを持っており, 計算結果が大きいニューロンを推測結果とする. 学習時はそれぞれの推測結果となる確率の形にするために, ソフトマックス関数に通すことが多い.

2.1.2　回帰問題

入力に対して, 連続的な予測数値を出力する問題である. 例えば, 大気質の予測では, 入力された数時間の大気質の成分データから, 次の時間に含まれる大気質の成分の数値を出力する. 回帰問題を解くニューラルネットワークは出力層に数値を予測したい要素の数だけニューロンを持っている. 分類問題とは違い, 学習時にソフトマックス関数に通されることはなく, 出力層のニューロンの値がそのまま出力される.

2.2　再帰型ニューラルネットワーク

2.2.1　再帰型ニューラルネットワークの原理

再帰型ニューラルネットワーク (Recurrent Neural Network, 以下 RNN) は音声認識や言語処理といった文脈のある時系列データを扱うニューラルネットワークである. 例えば 人の声の音声波形から性別を推定する問題がある場合, 図 1 の様に, 連続的な波形を短い時間間隔で離散化し, それぞれを $x_1, x_2, ...x_t$ とすることによって時系列データとして入力することが可能である.

再帰型ニューラルネットワーク (Recurrent Neural Network, 以下 RNN) は図 2 の様に内部に閉路を持つニューラルネットワークである. DNN などの従来のニューラルネットワークアーキテクチャと入力層, 中間層, 出力層を持つことは共通であるが, 中間層の出力が自身の入力に戻される帰還路を持つ.

RNN の動作の詳細について述べる. RNN は各時刻にひとつずつ入力を受け取る. 時刻 t で入力されたデータは従来のネットワークと同様に入力層から中間層へ伝搬される.

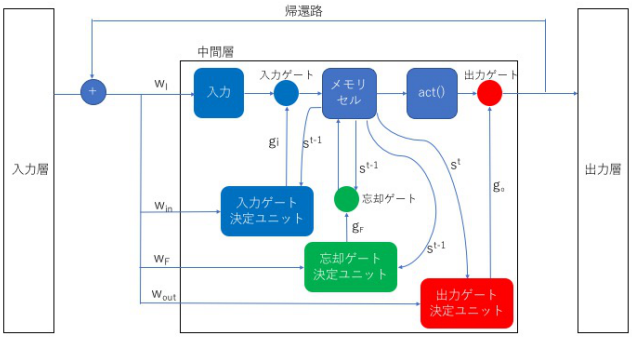

図 3　LSTM のメモリユニット

活性化関数を通した中間層の出力はそのまま出力層へ伝搬すると同時に, 中間層自身の入力へ帰還する. 出力層へ伝搬したものは時刻 t の予測結果として出力され, 中間層の入力へ帰還したものは時刻 $t+1$ の中間層への入力となる. これにより時刻 $t+1$ での出力は時刻 t 以前の中間層の出力の影響を受けることになり, 時系列データの文脈を捉えることが可能になる. ひとつの入力からひとつの出力への写像を近似する従来のネットワークとは異なり, RNN は系列から系列への写像を近似し, 時刻ごとに出力がある. RNN は分類問題と回帰問題の両方で利用されており, 特に株価予測のように連続値を予測する回帰問題も扱う [14].

2.2.2　長・短期記憶 (LSTM)

一般に, 層が深いニューラルネットワークでは勾配消失問題 (付録 A.1 参照) がある. RNN は層が浅くても中間層において再帰構造を持つため, 入力する時系列データが長くなると情報が伝搬していく層の数は深くなる. そのため, RNN では勾配消失問題が起こりやすく, 長期にわたる記憶を実現することは困難になる. 短期の記憶しか実現できない RNN の問題を踏まえて提案された方法が長・短期記憶 (Long Short-Term Memory, 以下 LSTM) である.

LSTM の基本的なアーキテクチャは RNN と同じであるが, 再帰構造を持つ中間層をメモリユニットと呼ばれる要素で置換している. メモリユニットを図 3 に示す.

LSTM ではこの様に複雑なアーキテクチャを持つため, 基

本的な RNN と比較して計算量が大きくなる．また, LSTM は入出力の形態によって, その型が異なる．例えば, 各時刻ごとに入力および出力がある Many to Many 型, 各時刻ごとに入力が与えられるが出力は最終時刻のみされる Many to One 型, 最初の時刻のみ入力をを与え各時刻ごとに出力がある One to Many 型がある．通常のニューラルネットワークはひとつの入力に対して, ひとつの出力をもつため, One to One 型といえる．各型の詳細は付録 A.2 に記載する．

2.3 モデル抽出攻撃の概要

モデル抽出攻撃の問題設定について述べる．まず, クラウド環境において訓練データ $D_1 \subset D$ を用いてモデルが訓練されているとし, この学習済みモデルを以下ではオリジナルモデルと呼ぶ．一般ユーザはクラウドへ料金を払って API クエリによって入力データを与える．この入力データがクラウド環境内のオリジナルモデルに入力され, オリジナルモデルが予測結果をユーザにレスポンスとして返すことによってサービスが成立する．

このとき, 一般ユーザの中に存在する悪意を持った者 (以下, 攻撃者) は, 自身が所有する訓練データの一部 $D_2 \subset D$ を入力データとして API を利用することで, オリジナルモデルが計算した予測結果を使って攻撃者自身が所有する学習モデルを訓練することが可能である．すなわち, 攻撃者はクラウドが持つ訓練結果や計算資源を踏み台にすることで, 自身の環境の中にクラウド環境内で動作するオリジナルモデルと同等以上の性能を持つモデルを得ることが可能となる．この攻撃手法をモデル抽出攻撃と呼び, 攻撃者が自身の環境内で所有する学習モデルを抽出モデルと呼ぶ．モデル抽出攻撃の直観を図 4 に示す．

モデル抽出攻撃の特徴は, 攻撃者の観点からはデータ収集および学習コストを大幅に削減したうえでモデルを得る点にある．一般にデータ収集および学習は負荷の高い作業であり, これらを経て得られたモデルはクラウドにとって重要な資産といえる．攻撃者はオリジナルモデルを踏み台にすることで, これらの負荷の高い作業を回避したまま, その資産ともいえるモデルを得ることが可能となる．とくに近年ではオリジナルモデルよりも高い精度を得られる抽出モデルについても文献 [4], [5] などで示されている．

3. 再帰型ニューラルネットワークへのモデル抽出攻撃

3.1 問題設定：RNN に対する場合

本節では本稿で取り扱う問題の特徴として, RNN に対してモデル抽出攻撃を行う場合の技術課題および制約条件について述べる．

計算資源　オリジナルモデルはクラウド上の潤沢な計算資源の元で実現しているが, モデル抽出攻撃を行う攻撃者は自身の計算資源の少ない環境で抽出モデルを実現させる必要がある．再帰型ニューラルネットワークにおける計算資源では, RNN よりも LSTM の方が資源が必要になる．すなわち, LSTM に対し RNN を用いて抽出モデルを実現する．

分類問題における入出力　再帰型ニューラルネットワークと, DNN の違いとして, 各時刻において入力と出力が存在する．これらの途中入力および途中出力情報は, 最終出力を計算する過程においてフィードバックをモデル内でかけるために利用される．再帰型ニューラルネットワークではこれらの途中入力および途中出力から, 攻撃者がモデル抽出攻撃に対する利点を得られるか議論する．

回帰問題における攻撃の実現　2.2 節でも述べたように, RNN は回帰問題に対し利用されることもあり得る．従来のモデル抽出攻撃は画像分類など分類問題を対象とした設定で行っており, とくにニューラルネットワークではソフトマックス関数を用いた分類が利用されている．しかしながら, 回帰問題を対象としたニューラルネットワークではソフトマックス関数が使えないことから, 例えば文献 [4] でされているようなソフトマックス関数に細工を施すことでクエリ数を削減する従来手法が利用できない．このため, 回帰問題に対してクエリ数を削減したまま精度を改善する方法を新たに考える必要がある．とくに, 一般に回帰問題では各中間値の出力は次の時間に入力に相当する（あるいはそもそも途中で出力を出さない）ことから, 攻撃者はオリジナルモデルからの最終出力のみを用いて精度の改善を考えなければならない．

本稿における RNN に対するモデル抽出攻撃では上述した計算資源の観点に加え, 分類問題と回帰問題それぞれの観点から, モデル抽出攻撃およびその精度への影響を評価する．

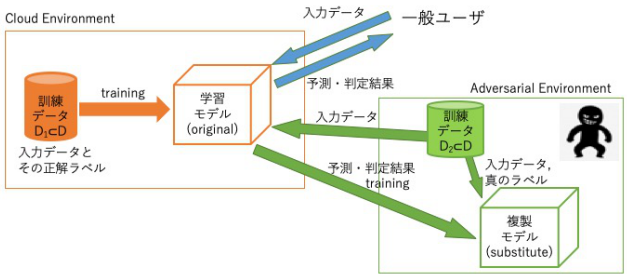

図 4　モデル抽出攻撃の概要

3.2 攻撃手法

本節では再帰型ニューラルネットワークの特性を利用し

たモデル抽出攻撃について説明する．以降では単純なアーキテクチャの再帰型ニューラルネットワークを RNN と記載し，長・短期記憶のためのアーキテクチャの再帰型ニューラルネットワークを LSTM と記載する．

2.2.2 節で述べた Many to Many 型 LSTM へのモデル抽出攻撃として，クラウド環境の訓練データを D_c，攻撃者環境の訓練データを D_a とする．このとき，D_c を用いてクラウド上に保存されているオリジナルモデルの LSTM を学習させる．次に攻撃者は D_a を用いて自身の抽出モデルとなる RNN を学習させるとともに，入力データをクラウドに投入して返ってきた推測結果をもとに，自身の RNN を学習させる．

以下にそれぞれの問題における詳細を述べる．

3.2.1 分類問題における攻撃方法

情報が漏れやすい時刻を把握し，その時刻の情報を集中的に用いて抽出を行う．具体的な攻撃の手順を以下に説明する．

一般に分類問題を解くニューラルネットワークでは出力層において推測時にはソフトマックス関数に通さずに出力しており，このソフトマックス関数を通す前の出力値を本稿では便宜上 logits と呼ぶ．logits は分類結果として各ラベルを確率付きで表現するソフトラベル（soft-label）である一方，クラウドおよび攻撃者が所持する訓練データの正解ラベルは正解がどのラベルであるかのみを表現するワン・ホット・ラベル（one-hot-label）である．図 6 にその概観を示す．このとき攻撃者の攻撃手順は，以下のように行われる．

(1) クラウド LSTM から情報が漏れやすい時刻の把握: 攻撃者は，オリジナルモデルから返ってくる時刻 0 から時刻 28 までの logits において，各ベクトル成分の最大値のインデックスを求め，正解ラベルと比較することで，それぞれの時刻における正解率 (精度) を求める．このとき，精度の高い時刻を**情報が漏れやすい時刻**とする．

(2) 情報が漏れやすい時刻の情報を用いた集中的な抽出: 前項で求めた情報が漏れやすい時刻の logits に対し，温度付きソフトマックス関数を用いる．このとき，学習はまず，自身が所持する D_a 内の入力データに対応する正解ラベル (ワン・ホット・ラベル) で損失関数を計算し，パラメータの更新を行う．その後に，logits に温度付きソフトマックス関数 [9] を通したソフトラベルを正解ラベルとして損失関数を計算することで，パラメータの更新を行う．

ここでいう温度付きソフトマックス関数は式 (1) で定義され，そのグラフは図 6 の様になる．$T = 1$ のときは通常のソフトマックス関数に一致する．直観として，温度 T に比例してグラフの勾配が滑らかになること，すなわち，学習の収束具合を早めることが可能となる．これにより少ないデータ数で高い精度の抽出モデルを得ることが可能となる．

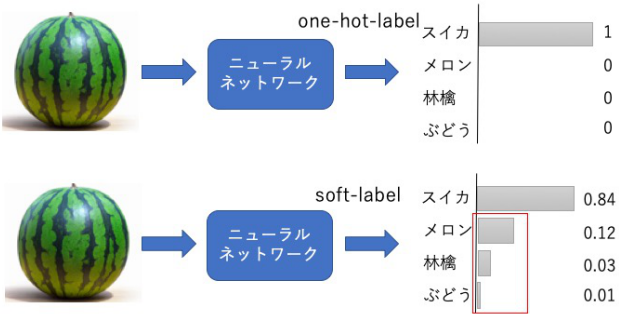

図 5 ワン・ホット・ラベル表現とソフトラベル表現

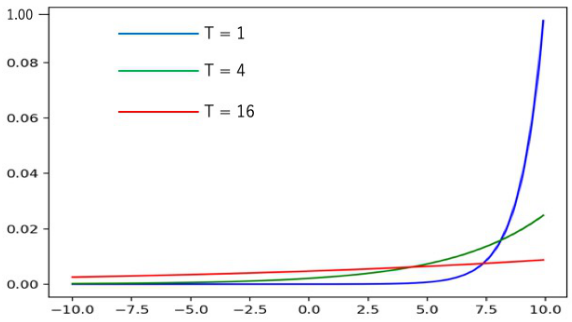

図 6 温度付きソフトマックス関数

以降では，温度付きソフトマックス関数を用いること，また，その温度を変化させることで，モデル抽出攻撃に対する影響も検証する．

$$softmax(k) = \frac{e^{\frac{a_k}{T}}}{\sum_{i=1}^{n} e^{\frac{a_i}{T}}} \tag{1}$$

3.2.2 回帰問題における攻撃方法

回帰問題を解くニューラルネットワークの場合，logits に対して温度付きソフトマックス関数は使えないため，L2 ノルムからなる損失関数 $L2loss$ を用いる．

ニューラルネットワークの蒸留 [10] では，L_2loss は式 2 の様に定義されている．

$$L_a = \frac{1}{2}||logits - prediction||_2^2 \tag{2}$$

しかしながら，この計算だけでは分類問題に対して行ったようなクエリ数の削減と精度の向上の両立が期待できない．これは単に logits と予測値の差分を計算しているだけだからである．このため，抽出モデルの出力 R_s と正解値 y の L_2loss がオリジナルモデルの出力 R_t と正解値 y の L_2loss よりもある程度大きくなるときのみ，パラメータの更新を行い，それ以外は損失関数を 0 として扱う方法 [11] を用いることで，攻撃者が学習の幅を調整できる仕組みを導入する．具体的には式 (3) で損失関数を定義する．

$$L_{b0} = \begin{cases} ||R_s - y||_2^2, if||R_S - y||_2^2 + m > ||R_t - y||_2^2 \\ 0, otherwise \end{cases} \tag{3}$$

m は各種データセットから期待できる予測値に対して微小な数値である.

また, 損失関数による過剰な影響を防ぐため, 式 (3) から二乗計算を外した損失関数も式 (4) で定義する.

$$L_b = \begin{cases} ||R_s - y||_2, if ||R_S - y||_2 + m > ||R_t - y||_2 \\ 0, otherwise \end{cases} \quad (4)$$

これらの損失関数 L_{b0}, L_b のいずれかを用いることで, 抽出モデルの出力と正解値の差が大きいときのみ更新を行う形式での収束の調整が期待できる. 次節以降の実験評価では, これらの損失関数 L_a, L_{b0}, L_b をそれぞれ用いてモデル抽出攻撃を行う.

4. 実験評価

本節では前節で述べた RNN に対するモデル抽出攻撃に対し実験を行い, 精度の観点から有効性を評価する. とくに分類問題および回帰問題それぞれにおいて実験することで, ニューラルネットワーク構造ごとの影響を含めて評価する.

4.1 実験設定

使用した実験環境を表 1 に示す. また, ニューラルネットワークの学習方法は, Adam Optimizer を学習率 0.001 で用いる.

4.1.1 分類問題における設定

まず, Many to Many 型 LSTM へのモデル抽出攻撃の実験として MNIST データセット[*3]を扱う. MNIST データセットは訓練データ 55000 枚, テストデータ 11000 枚で構成されており, 1 枚あたり 28 × 28 ピクセルで 0 から 9 の手書き文字が表現されている. 図 7 は MNIST データセットのイメージ画像である.

今回は MNIST データセットを時系列データに変換して扱う. 具体的には, 図 8 の様に 28 × 28 ピクセルの入力データを行ごとに短冊状に分割し, t 行目を時刻 t のデータとする. 入力データが 28 行あることから, 今回は時刻 1 から時刻 28 の時系列データとなる. また, 各時刻ごとに予測結果を出力することから Many to Many 型 LSTM となる.

手書き数字を分類するためのニューラルネットワーク構造は, 文献 [4] において DNN と CNN が扱われている. 文献 [4] のモデル抽出攻撃の精度と比較するため, 実験設定を統一する. 具体的には, 図 9 の様に, MNIST データセットの訓練データ 55000 個を 11000 個ずつに分割し, それぞれを D_1, D_2, D_3, D_4, D_5 とし, 四つをクラウド環境での訓練データ (D_c) に, 残りの 1 つを攻撃者環境での訓練データ (D_a) にする. D_a の選び方は 5 通りあるため, 5 通り実験を行い, それらの平均を本実験の結果とする. MNIST の場合は分類問題となるため, 温度付きソフトマックス関数を通したクロスエントロピー誤差を損失関数として用いる. 温度付きソフトマックス関数の温度は $T = 1, 4, 16$ で評価する. 具体的には, 抽出モデルを正解ラベルと通常のクロスエントロピー誤差を用いて学習させた後に, クラウドへのクエリによって得た logits を温度付きソフトマックス関数に通し, クロスエントロピー誤差を損失関数として学習させる.

表 1 実験環境

開発プラットフォーム	Tensor Flow 2.0.
OS	Ubuntu 18.04
GPU	NVIDIA Tesla K80 12GB
メモリ	13GB RAM
ストレージ	360GB

[*3] http://yann.lecun.com/exdb/mnist/

図 7 MNIST データセット

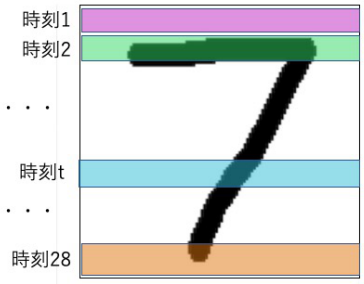

図 8 MNIST の時系列データへの変換

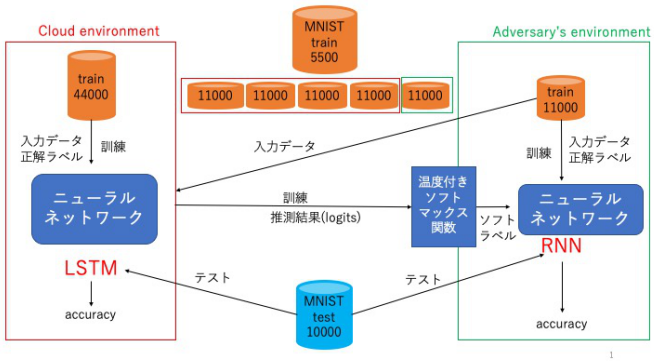

図 9 構築したクラウド環境と攻撃者環境 (MNIST データセット)

表 2　学習設定 (分類問題)

エポック数	220
1 エポック内反復数	$\frac{\lvert D_a \rvert}{50}$
バッチサイズ	50

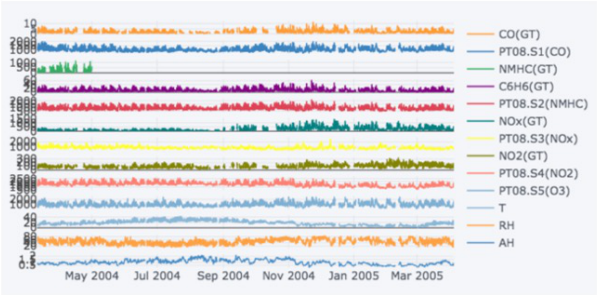

図 10　Air Quality データセット

分類問題における評価の観点は, 以下の通りである.

(1) 漏れやすい時刻の調査
(2) 抽出モデル構造による精度の違い
(3) 攻撃者が所持する訓練データ数割合による精度の違い
(4) 温度付きソフトマックス関数の温度による精度の違い

精度評価はクラウド環境の LSTM も攻撃者環境の RNN も MNIST データセットのテストデータ 10000 個を用いて行う. 今回, クラウド上で稼働するオリジナルモデルとして作成した LSTM の精度は 97.3 %であり, この精度を抽出モデルの精度の目標値とする. 実験にて用いる学習設定を表 2 に示す.

4.1.2　回帰問題における設定

回帰問題における実験として, 大気質に含まれる物質の量や温度などをセンサーデバイスによって収集して構成された Air Quality データセット[*4]を扱う. Air Quality データセットは　2004 年 3 月 10 日 18:00 から 2005 年 4 月 4 日 14:00 まで 1 時間おきに計測される大気質の成分数値のデータセットであり, それぞれの時間で物質の量や温度など 13 種類の値によって構成されている. 図 10 は Air Quality データセットのイメージ画像である.

Air Quality データセットの中から T(温度), AH(絶対湿度), CO(一酸化炭素に対する酸化スズの時間平均値), $NMHC$(非メタン炭化水素に対する酸化チタンの時間平均値), NO_x(窒素酸化物に対する酸化タングステンの時間平均値), NO_2(二酸化窒素（NO2）に対する酸化タングステンの時間平均値) の 6 種類の値を用いる. 図 11 に示すように, それぞれの再帰型ニューラルネットワークは 72 時間分の 6 種類の計測値を 72 時間分の時系列データとして入力し, 73 時間目の 6 種類の計測値を予測して出力する[*5].

2005 年のデータをテストデータとし, 2004 年のデータをクラウド環境での訓練データ (D_c) とする. 攻撃者は 2004 年の 4 月から 12 月までの 9 ヶ月間のうち, n 月分 ($n = 1, 3, 6$) のデータ D_a を訓練データとして持っているとする. D_a の選び方は複数あるため, 表 3 に従って実験し, 平均をとる.

[*4] https://archive.ics.uci.edu/ml/machine-learning-databases/
[*5] https://deepinsider.jp/tutor/introtensorflow/buildrnn

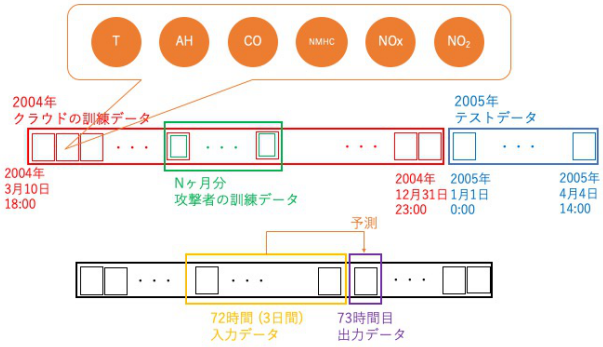

図 11　構築したクラウド環境と攻撃者環境 (Air Quality データセット)

表 3　攻撃者が所持する訓練データの組合せ

訓練データの規模	実験に用いた訓練データの月
1 ヶ月分	{4}, {10}
3 ヶ月分	{4, 5, 6}, {7,8, 9}
6 ヶ月分	{4, 5, 6, 7, 8, 9}, {7, 8, 9, 10, 11, 12}

表 4　学習設定 (回帰問題)

エポック数	$45n$
1 エポック内反復数	$\frac{\lvert D_a \rvert}{16}$
バッチサイズ	16

オリジナルモデルの学習を行う際は, D_C における正解値と予測値の平均絶対誤差を損失関数として用いる. また, 抽出モデルの損失関数について, D_a における正解値と予測値の平均絶対誤差, L_a, L_{b0}, L_b の 4 種類それぞれ用いる. すなわち, ひとつ目の場合はクラウドへのクエリ結果から得た情報を利用しないが, 他の 3 つに関してはその情報を利用する.

回帰問題における評価の観点は, 以下の通りである.

(1) 抽出モデル構造による精度の違い
(2) 攻撃者が所持する訓練データ数割合による精度の違い
(3) L_2 ノルムを用いた損失関数の効果

今回, クラウド上で稼働するオリジナルモデルとして作成した LSTM の精度は 94.5 %であり, この精度を抽出モデルの精度の目標値とする. 実験にて用いる学習設定を表 4 に示す.

4.2　実験結果

4.2.1　分類問題における実験結果

まず, 情報が漏れやすい時刻の確認結果を図 12 に示す. 図 12 の結果によると, LSTM では時刻 21 から精度が増加し始め, 時刻 26 以降は 90 %を超えている. 一方, クラウド

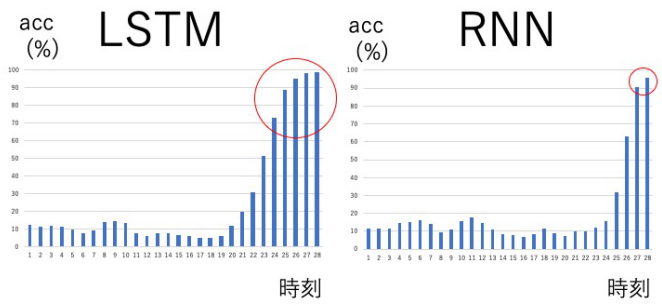

図 12　情報が漏れやすい時刻

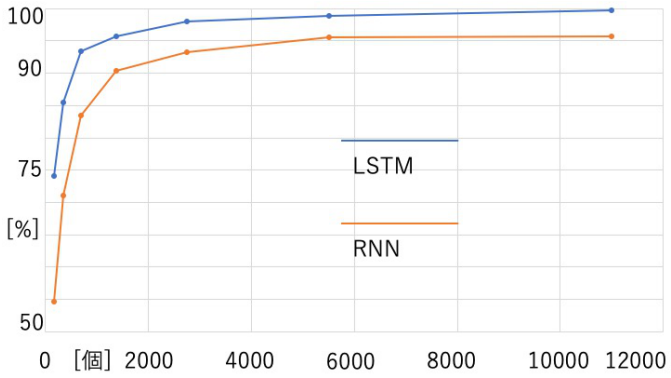

図 14　クエリ数による抽出モデルの精度の変化

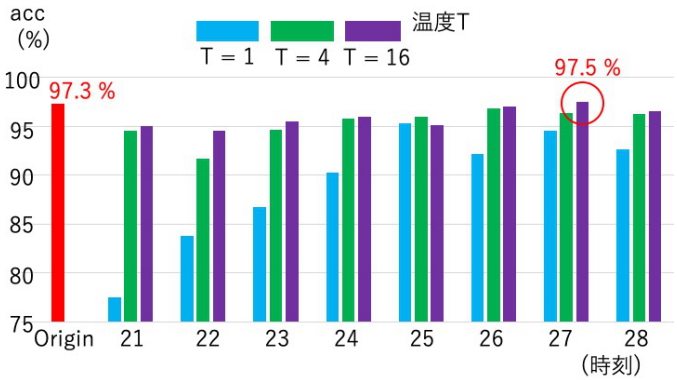

図 13　情報が漏れやすい時刻での集中的な抽出結果

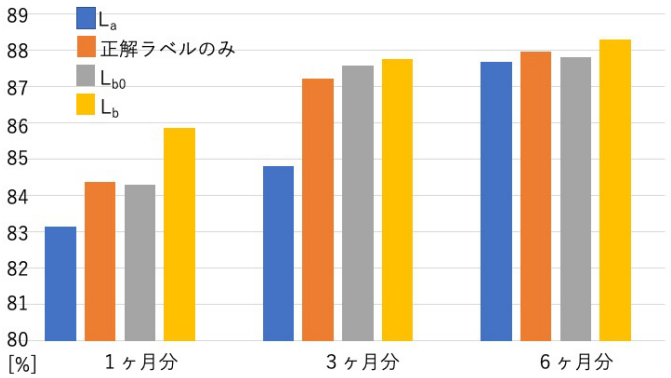

図 15　Air Quality でのモデル抽出攻撃の結果 (L_{b0} と L_b は $m = 5$)

で LSTM の代わりに RNN を訓練した追加実験では時刻 25 から精度が増加し始め, 時刻 27 以降で 90 %を超える. 故に, LSTM の方が RNN よりも最終結果の精度が大きいことに加え, 早い段階で大きな情報量が漏れることがわかる.

次に LSTM の時刻 21 以降を集中的に抽出した結果を図**図 13** に示す. これにより, 時刻 21 でさえ 95 %の精度のモデルを抽出できていることが分かる. とくにに温度 $T = 16$ とした温度付きソフトマックス関数では時刻 27 において精度が 97.5 %となり, オリジナルモデルの精度である 97.3 %を上回る結果となった. なお, クラウド上のモデルが時刻 28 の最終予測結果のみ出力をする Many to One 型 LSTM の場合, 攻撃者は最終結果しか知ることはできず, この 97.5%という精度のモデルを抽出することはできない.

最後に時刻 27, 温度 $T = 16$ において, クエリ数によって抽出モデルの精度がどのように変化するかを調べた. 結果を図 14 に示す. これにより, 攻撃者が所持する訓練データ数が少なくなれば, 急激に抽出モデルの精度も下がることが分かる.

4.2.2　回帰問題における実験結果

Air Quality を用いた実験結果を図 15 に示す. これにより, 攻撃者が所持する訓練データで学習 (平均絶対誤差を利用) した抽出モデルよりも L_a を損失関数として抽出したモデルの精度の方が小さくなり, L_{b0}, L_b を損失関数として抽出したモデルの精度の方が大きくなることが確認された. L_b を用いた抽出では攻撃者が所持する訓練データの規模が, 1 ヶ月分では 85.9 %, 3 ヶ月分では 87.8 %, 6 ヶ月分では 88.3 % のモデルを抽出することができた.

5.　考察

5.1　分類問題における考察

LSTM はより長期間の記憶を保持するため, 情報が漏れる時刻の範囲が広がると考えられる. また, 時刻 21 で高精度のモデルを抽出することができた理由は, 数字の画像データの下の方は情報が少なく, 21 行目の段階で学習モデルが十分な知識を取り出せたことによるものであると考えられる. 本実験では MNIST を時系列データに変換することによって, 情報が漏れる時間帯を把握することができたが, 他の時系列データでは初期段階で情報を漏らす可能性もある. その場合は抽出が容易になり, より厳しい API 制限が必要になると考えられる.

更なる考察に向けた追加実験として, 抽出モデルに LSTM を用いた場合も行なっている. 抽出モデルに RNN を用いた場合は, 1375 個の訓練データ量で 90.3%の精度になる一方, 抽出モデルに LSTM を用いた場合は, 約 500 個の訓練データで同等の精度が得られている. また, LSTM 抽出モデルでは 110000 個の訓練データを用いた場合, 99.69% の精度のモデルが抽出できた. これにより, 再帰型ニューラ

ルネットワークにおいて, LSTM 抽出モデルでは RNN 抽出モデルより少ないクエリ数で高精度のモデルが抽出できることが確認された. これは岡田ら [4] による画像分類学習モデルへのモデル抽出攻撃の結果と共通点が見られる. 具体的には, 抽出モデルに DNN , CNN を用いた場合, それぞれ, 684 個, 171 個の訓練データの量で抽出モデルの精度が 90% 以上という結果である. これは, 画像分類学習においては, 抽出モデルに DNN よりも CNN を用いた方が, より少ないクエリ数で高精度のモデルが抽出できることを示していた. 本稿では, 時系列データ学習において抽出モデルに RNN よりも LSTM を用いた方が, より少ないクエリ数で高精度のモデルを抽出できることが示されている.

また, 本稿の実験では, 正解ラベルとソフトラベルによる抽出モデルの学習を別々に行なったが, 文献 [11] において, 両者を一度に用いて損失関数を定義しており, この関数を用いることで学習回数 (エポック数) の削減が期待できる. 具体的には式 (5) で損失関数を定義する. P_s は抽出モデルの予測値, P_t はクラウドの予測値, μ はハイパーパラメータである. L_{hard} は正解ラベル y によって得られた損失関数の値, L_{soft} はソフトラベルによって得られた損失関数の値を表している.

$$L = \mu L_{hard}(P_s, y) + (1-\mu)L_{soft}(P_s, P_t) \qquad (5)$$

適切な μ を設定することによって抽出モデルの精度収束に対して高い効果があると考えられる.

本稿の実験では中間値の出力を時刻ひとつ分のみ用いて精度の高いモデルを抽出したが, 複数の中間値を組み合わせたり, 中間値と最終出力値の情報を組み合わせる方法も効果があると期待できる. ニューラルネットワークの蒸留において教師モデルの中間値の情報を用いて生徒モデルを学習させる Hint Learning という手法 [15] があり, 教師モデルの中間出力と生徒モデルの入力の間に追加の層を導入することによって生徒モデルの学習と精度向上を高めることが可能である [11]. RNN の中間出力にの情報に対して Hint Learning を用いる実証実験は今後の課題である.

5.2 回帰問題における考察

式 (2) で定義される L_2loss のみでは十分なデータセットを用いることができる蒸留では役に立つが, モデル抽出攻撃のように少ないデータセットのもとでは効果が得られないと考えられる. 具体的には, 温度付きソフトマックス関数ではその出力値が確率になることから $[0, 1]$ の範囲をとることに対し, L_2 ノルムではその値が発散しやすいため, 損失関数が余剰に大きくなってしまう. これにより, できるだけクエリ数や, エポック数を小さくするというモデル抽出攻撃において, 抽出モデルの学習が収束できなかったと考えられる. この問題を解決したのが, 式 (4) で定義される L_2loss である. 抽出モデルの出力と正解値の L_2loss がオリジナルモデルの出力と正解値の L_2loss よりもある程度大きくなるときのみ, パラメータの更新を行い, それ以外は損失関数を 0 として扱うことで, 収束を調整することが可能となっていた. L_{b0} と L_b において, m の値を変化させると図 16 の様になる. $m = 100$ の際に精度が低下し, $m = 500$ において, 過学習が起きていると考えている. 最適な m を見つけることが必要である.

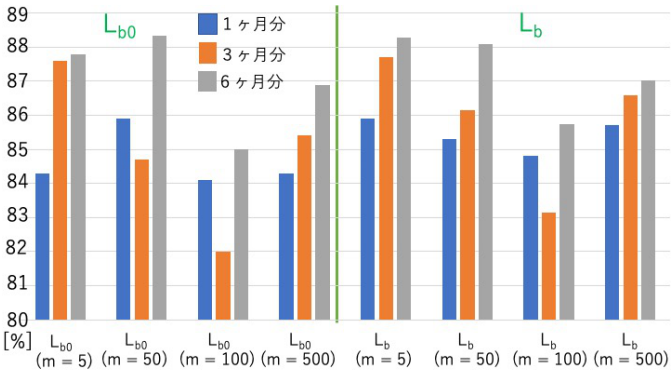

図 16 m を変化させて L_{b0} と L_b を利用した時の抽出モデルの精度

また, 同じ訓練データ量でもどこの月を用いるかで精度に差があり, これは現状の実験では学習が収束していないことに起因する可能性もある. すなわち, 学習回数を増やしていくことで L_{b0}, L_b を用いた損失関数によって, より高い精度を得られる可能性がある. 本実験結果の傾向から, 十分な訓練データ数, 学習回数の元では, どの損失関数を用いても同じ精度に収束していくと考えられる.

これらの予測に関する実証実験は今後の課題である.

6. 結論

モデル抽出攻撃は学習モデルへのクエリを通じてモデルを抽出するものであり, 従来研究では単純なニューラルネットワークでのみ攻撃が検証されていた. MNIST データセットを時系列データに変換した今回の実験では, RNN に対してモデル抽出攻撃を検証し, 正解ラベルによる学習で時間をしぼった後, 予測ラベルで学習することで効率的に攻撃ができることを示した. とくに, 予測モデルからの学習では温度付きソフトマックス関数を用いることで, オリジナルモデルよりも高い予測精度のモデルを抽出できた. また, LSTM は RNN と比べ情報が漏れやくく, より厳しい API 制限が必要であることが分かった. さらに抽出モデルの損失関数の定義次第で, 本稿の実験で得られた精度と同等のモデルをより少ないエポック数で得られることも考えられる. なお, 今回は回帰問題に対してはオリジナルモデル以上の精度を持つモデルを得ること自体はできなかったが, これは損失関数におけるパラメータの調整で改善できる可能性はある. この回帰問題における有効なパラメータの模索は今後の課題である.

謝辞

本研究開発の一部は，総合科学技術・イノベーション会議の戦略的イノベーション創造プログラム（SIP）第 2 期「IoT 社会に対応したサイバー・フィジカル・セキュリティ」（管理法人：NEDO）にて実施されております．

参考文献

[1] Yann LeCun, Yoshua Bengio and Geoffrey Hinton: Deep learning, *Nature volume 521*, pp. 436–444 (2015)

[2] Tramér, F., Zhang, F., Juels, A., K., Reiter, M. and Ristenpart, T.: Stealing Machine Learning Models via Prediction APIs, *USENIX Security*, pp. 601–618 (2016).

[3] Binghui Wang, Neil Zhenqiang Gong: Stealing Hyperparameters in Machine Learning, *IEEE S&P*, pp. 36–52 (2018).

[4] 岡田莉奈，長谷川聡：画像分類深層学習器に対する Model Extraction 攻撃の検証，*CSS*, pp. 201–208 (2018).

[5] Mika, J., Sebastian, S., Alexey, D., Samuel, M. and N., A.: PRADA: Protecting against DNN Model Stealing Attacks, *EuroS&P*, (online), available from ⟨http://arxiv.org/abs/1805.02628⟩ (2019).

[6] Nedim, S. and Pavel, L.: Practical Evasion of a Learning-Based Classifier: A Case Study, *IEEE S&P*, pp. 197–211 (2014).

[7] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B. and Swami, A.: Practical Black-Box Attacks Against Machine Learning, *Asia CCS*, pp. 506–519 (2017).

[8] Kesarwani, M., Mukhoty, B., Arya, V. and Mehta, S.: Model Extraction Warning in MLaaS Paradigm, *ACSAC*, pp. 371–380 (2018).

[9] Hinton, G., Vinyals, O. and Dean, J.: Distilling the knowledge in a neural network, (online), available from ⟨https://arxiv.org/abs/1503.02531⟩ (2015).

[10] Jimmy, Ba, L. and Caruana, R.: Do Deep Nets Really Need to be Deep?, *NIPS*, (online), available from ⟨https://arxiv.org/abs/1312.6184⟩ (2014).

[11] Chen, G., Choi, W., Yu, X., Han, T. and Chandraker, M.: Learning Efficient Object Detection Models with Knowledge Distillation, *NIPS*, (online), available from ⟨https://papers.nips.cc/paper/6676-learning-efficient-object-detection-models-with-knowledge-distillation.pdf⟩ (2017).

[12] Bucila, C., Caruana, R. and Niculescu-Mizil, A.: Model Compression, *KDD*, pp. 535–541 (2006).

[13] Naghibijouybari, H., Neupane, A., Qian, Z. and Abu-Ghazaleh, N.: Rendered Insecure: GPU Side Channel Attacks are Practical, *CCS*, pp. 2139–2153 (2018).

[14] 松井 藤五郎 , 汐月 智也: LSTM を用いた株価変動予測, *JSAI* (2017).

[15] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio.: Fitnets: Hints for thin deep nets., (online), available from ⟨https://arxiv.org/abs/1412.6550⟩ (2014).

付　　録

A.1　勾配消失問題

ニューラルネットワークには順伝搬の過程で非線形である活性化関数を通しており，出力が過大に発散することはない．例えば，ソフトマックス関数を通した出力は [0, 1] の範囲に収まる．一方で，学習時の勾配の計算は線形である微分の計算を繰り返すため，各層の重みが大きい場合や，層が深い場合は，急速に発散したり，急速に消失してしまう可能性がある．これを勾配消失問題という．

A.2　LSTM の型

A.2.1　Many to Many 型 LSTM

図 A·1 の様に，各時刻ごとに観測情報の入力があり，各時刻ごとに予測結果となる出力がある LSTM．時系列データを入力とする分類問題において，各段階で最良な予測結果を出力する形で用いられる．

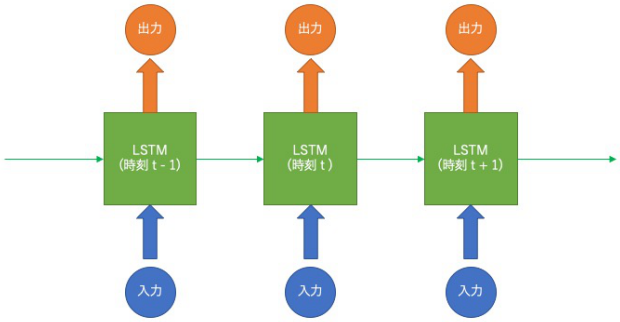

図 A·1　Many to Many 型 LSTM

A.2.2　Many to One 型 LSTM

図 A·2 の様に，各時刻ごとに観測情報の入力があり，最終時刻の入力後に予測結果がひとつのみ出力される LSTM．具体的には，音声データを時刻ごとに入力していき，声の持ち主の性別を最後に判定する処理などで用いられる．

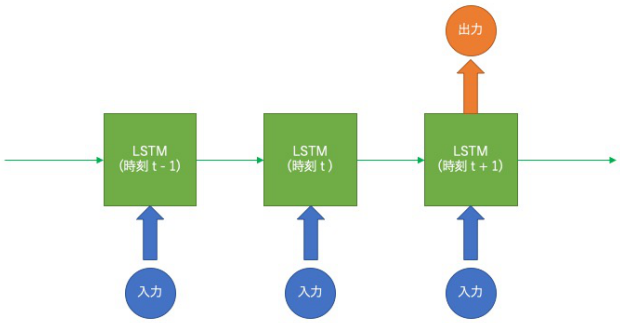

図 A·2　Many to One 型 LSTM

A.2.3　One to Many 型 LSTM

図 A·3 の様に，最初のステップにのみ，観測情報の入力があり，それ以降，各時刻ごとに予測結果を出力していく LSTM．モデルの出力自体が次の時刻の入力となっていく場合もある．具体的には初期位相を与えられた状態で sin

波全体を予測していく処理などで用いられる.

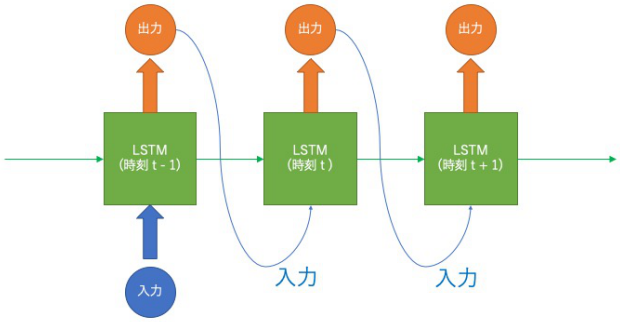

図 **A·3** One to Many 型 LSTM