# Binarized Neural Networksを用いた秘匿予測プロトコル

西田 直央[1]　大庭 達海[2]　加藤 遼[1]　海上 勇二[1]　山田 翔太[3]　アッタラパドゥン ナッタポン[3]
照屋 唯紀[3]　松田 隆宏[3]　花岡 悟一郎[3]

**概要**：ユーザのプライバシ情報を秘匿したまま，ニューラルネットワークのデータ処理を行う技術（Secure Neural Network, SNN）が研究されている．SNN では暗号化したまま大量の数値計算を行うため，計算量が膨大となることが知られている．そこで本研究では，SNN の予測処理に関する計算量を削減するため新たな手法を提案する．提案手法では，ニューラルネットワークに Binarized Neural Networks を用いることで，SNN の秘匿対象を短いビット長で表現し，計算量を削減することが可能である．本提案では，高い予測精度を達成可能であり，安全で効率的な SNN の実現が見込まれる．

**キーワード**：秘密分散法，マルチパーティ計算，機械学習，ニューラルネットワーク, Binarized Neural Networks

# Private Machine Learning Classification Based on Binarized Neural Networks

Naohisa Nishida[1]　Tatsumi Oba[2]　Ryo Kato[1]　Yuji Unagami[1]　Shota Yamada[3]
Nuttapong Attrapadung[3]　Tadanori Teruya[3]　Takahiro Matsuda[3]　Goichiro Hanaoka[3]

***Abstract:*** Secure Neural Networks (SNN), which enables computation of neural network while preserving the privacy of user information, has been studied. It is known that the computational cost of SNN is very expensive because it performs a large number of numerical computations in encrypted forms. In this work, we propose a new scheme that reduces the computational cost of the prediction phase in SNN. In the proposed scheme, we use Binarized Neural Networks instead of the conventional neural network. This allows us to represent secret information by short bit strings and therefore reduce the communication cost. The proposed scheme achieves high prediction accuracy and is secure and efficient.

***Keywords:*** Secret Sharing, Multi Party Computation, Machine Learning, Neural Networks, Binarized Neural Networks

## 1. はじめに

### 1.1 背景

近年，様々な企業が機械学習を利用したサービスを展開している．特に機械学習技術の一つであるニューラルネットワークは，予測精度の高さや，応用分野の広さから着目されている手法の一つである．

また，機械学習の利用が想定されるケースとして，学習に用いるデータや学習済みの予測モデルの所有者と，予測モデルを利用したいと考えるユーザが異なるケースはしばしば起こり得る．例えば，サービス提供者が学習用データを，サービスのユーザが予測モデルに入力したいデータをそれぞれ所持しており，外部のクラウド上で学習や予測を行う場合がある．これらのデータにはユーザのプライバシ情報が含まれていることも多いため，互いに情報を共有することなく，学習や予測を行う技術が求められている．

この問題を解決するため，ユーザのプライバシ情報を秘匿したままニューラルネットワークを用いて学習及び予測

1　パナソニック株式会社　ビジネスイノベーション本部
Business Innovation Division, Panasonic Corporation
2　パナソニック株式会社　製品セキュリティセンター
Product Security Center, Panasonic Corporation
3　国立研究開発法人 産業技術総合研究所 情報技術研究部門
National Institute of Advance Industrial Science and Technology, Information Technology Research Institute

を行う技術の研究開発が進められている．しかし従来手法では，計算量が膨大となることや，通常のニューラルネットワークに比べて予測精度が低下するという問題がある．

そこで我々は，ユーザのプライバシ情報を秘匿しつつ，ニューラルネットワークの予測処理に関する計算量を削減し，高い予測精度の達成が見込まれる新たな手法を提案する．

### 1.2 関連研究

Barni ら [1] は，加法準同型暗号及び GarbledCircuit を用いた，ニューラルネットワークの予測処理の秘匿計算プロトコルを初めて提案した．しかし，Barni らのプロトコルは，計算途中において出力情報以上の情報が漏洩してしまう．Orlandi ら [8] は，Barni らの方式の問題を解決した方式を提案したが，非常に効率が悪い．Dowlin ら [5] は，多項式計算をサポートする準同型暗号に基づき，Crypto-Net と呼ばれるニューラルネットワークの予測処理の秘匿計算プロトコルを提案した．このプロトコルは，準同型暗号を用いているため，通信回数は多くないが，非線形計算を要する中間層の数が増えた場合，予測精度が非常に早く低下する．Chabanne ら [4] は，BatchNormalization を用いることで，Dowlin らのプロトコルの予測精度を改良した．最近では，Mohassel ら [7] が，秘密分散法と GarbledCircuit を用いたニューラルネットワークの予測処理の秘匿計算プロトコルを提案した．彼らの論文では学習処理に焦点が当てられているが，予測処理も示されている．ニューラルネットワーク以外の機械学習の予測処理に関しては，Bost ら [2] が，超平面識別器などに基づく予測処理の効率的な秘匿計算プロトコルを提案している．

## 2. ニューラルネットワーク

本稿では，フィードフォワードニューラルネットワークと呼ばれる単純な構造のニューラルネットワークを，ニューラルネットワークと呼ぶ（図 **1**）．

ニューラルネットワークは，単純な線形演算と非線形演算を階層的に行うことで，複雑な非線形関数を近似可能な数理モデルである．

ニューラルネットワークは「学習」と「予測」の 2 つのフェーズに分解される．学習フェーズでは，教師データと呼ばれる入力（画素値など）と，入力に対応するラベル（画像のクラスを表す ID) のペアが複数与えられ，ニューラルネットワークが入力に対して適切なラベルを出力するように予測モデルのパラメータを更新する．

予測フェーズでは，ラベルが未知の入力データをニューラルネットワークに入力することで，ニューラルネットワークが予測するラベルを得る．

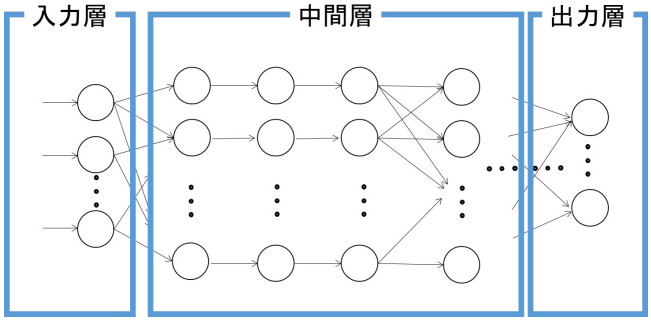


図 1　ニューラルネットワーク

### 2.1 表記

- $L$：ニューラルネットワークの層の数
- $n_i$：$i$ 層目のノード数 $(i = 0, 1, \ldots, L)$
- $\boldsymbol{a}^{(j)}$：ベクトル $\boldsymbol{a}$ の $j$ 要素目．
- $\boldsymbol{A} \times \boldsymbol{B}$：行列 $\boldsymbol{A}, \boldsymbol{B}$ の積．
- ReLU$(x)$：$x$ が正か 0 であれば $x$ を，負であれば 0 を返す関数．
- Sign$(x)$：$x$ が正か 0 であれば 1 を，負であれば –1 を返す関数．
- Argmax$(\boldsymbol{a})$：ベクトル $\boldsymbol{a}$ の要素のうち，最大値をとる要素のインデックスを返す関数．

特に言及がない限り，ベクトルは横ベクトルとする．

### 2.2 プロトコル

ニューラルネットワークの予測プロトコルの一つを Protocol 1 に示す．ニューラルネットワークは，FullConnection，BatchNormalization，Activation を複数回繰り返し，最後に Argmax を実行することで予測処理を行う．

**Protocol 1** Neural Networks

**Input:** $\boldsymbol{a}_0 \in \mathbb{R}^{n_0}$：入力ベクトル
$\boldsymbol{W}_i \in \mathbb{R}^{n_{i-1} \times n_i}$：重み行列．
$\boldsymbol{\theta}_i = (\boldsymbol{\gamma}_i, \mu_i, \boldsymbol{\beta}_i, \boldsymbol{\sigma}_i) \in (\mathbb{R}^{n_i})^4$：BatchNormalization パラメータ
**Output:** $index \in \{1, \cdots, n_L\}$：予測結果
**Procedure:**
1: **for** $i = 1$ to $L - 1$ **do**
2: $\quad \boldsymbol{f}_i \leftarrow$FullConnection$(\boldsymbol{a}_{i-1}, \boldsymbol{W}_i, n_{i-1}, n_i)$
3: $\quad \boldsymbol{b}_i \leftarrow$BatchNormalization$(\boldsymbol{f}_i, \boldsymbol{\theta}_i, n_i)$
4: $\quad \boldsymbol{a}_i \leftarrow$Activation$(\boldsymbol{b}_i)$
5: **end for**
6: $\boldsymbol{f}_L \leftarrow$FullConnection$(\boldsymbol{a}_{L-1}, \boldsymbol{W}_L, n_{i-1}, n_i)$
7: $\boldsymbol{b}_L \leftarrow$BatchNormalization$(\boldsymbol{f}_L, \boldsymbol{\theta}_L, n_i)$
8: $index \leftarrow$Argmax$(\boldsymbol{b}_L, n_i)$
9: Output $index$

FullConnection のプロトコルを Protocol 2 に示す．このプロトコルは，ベクトル $\boldsymbol{a}$ と学習した重み行列 $\boldsymbol{W}$ に対し，行列積 $\boldsymbol{a} \times \boldsymbol{W}$ を計算する．

**Protocol 2** FullConnection

**Input:** $n, m \in \mathbb{Z}$：ベクトルサイズ
$\boldsymbol{a} \in \mathbb{R}^n$
$\boldsymbol{W} \in \mathbb{R}^{n \times m}$：重み行列
**Output:** $\boldsymbol{f} \in \mathbb{Z}^m$：$\boldsymbol{a}$ と $\boldsymbol{W}$ の行列積
**Procedure:**
1: $\boldsymbol{f} \leftarrow \boldsymbol{a} \times \boldsymbol{W}$；行列積
2: Output $\boldsymbol{f}$

BatchNormalization のプロトコルを Protocol 3 に示す．このプロトコルは，ニューラルネットワークの学習時のノードの値の分布を均一化する技術であり，学習の高速化，予測の高速化につながる．

**Protocol 3** BatchNormalization

**Input:** $n \in \mathbb{Z}$：ベクトルサイズ
$\boldsymbol{f} \in \mathbb{R}^n$
$\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\mu}, \boldsymbol{\beta}, \boldsymbol{\sigma}) \in (\mathbb{R}^n)^4$：BatchNormalization パラメータ
$\epsilon$：小さい正の定数
**Output:** $\boldsymbol{b} \in \mathbb{R}^n$
**Procedure:**
1: **for** $j = 1$ to $n$ **do**
2: $\quad \boldsymbol{b}^{(j)} \leftarrow \frac{\boldsymbol{f}^{(j)}\boldsymbol{\gamma}^{(j)}}{\sqrt{\boldsymbol{\sigma}^{(j)^2}+\epsilon}} + \boldsymbol{\beta}^{(j)} - \frac{\boldsymbol{\mu}^{(j)}}{\sqrt{\boldsymbol{\sigma}^{(j)^2}+\epsilon}}$
3: **end for**
4: Output $\boldsymbol{b}$

Activation のプロトコルを Protocol 4 に示す．このプロトコルは，BatchNormalization が適用されたベクトル $\boldsymbol{b}$ に対し非線形処理を加える．Protocol 4 では非線形処理として ReLU を用いているが，他にも Sigmoid 関数やステップ関数が用いられる場合もある．

**Protocol 4** Activation

**Input:** $n \in \mathbb{Z}$：ベクトルサイズ
$\boldsymbol{b} \in \mathbb{R}^n$
**Output:** $\boldsymbol{a} \in \mathbb{R}^n$
**Procedure:**
1: **for** $j = 1$ to $n$ **do**
2: $\quad \boldsymbol{a}^{(j)} \leftarrow$ReLU$(\boldsymbol{b}^{(j)})$
3: **end for**
4: Output $\boldsymbol{a}$

ニューラルネットワークでは $\boldsymbol{b}_L$ のうち最大のノードのインデックスを予測結果として出力する．最終的な予測結果を出力するため，Argmax を用いて与えられたベクトル $(\boldsymbol{b}_L)$ から最大となる要素を探索し，そのインデックスを出力する．

### 2.3 Binarized Neural Networks

Binarized Neural Networks (BNN) は，重み行列の値と Activation 関数の出力の値が 1 または −1 に二値化されるニューラルネットワークである [6]．重み行列と Activation 関数を二値化することで，従来のニューラルネットワークに比べて必要なストレージサイズを削減することが可能になる．また，二値化したことで FullConnection 等の処理をビット演算で置き換えることが出来，従来手法よりも高速に動作する．

BNN の予測プロトコルを Protocol 5 に示す．このプロトコルは，従来のニューラルネットワークと同じく，FullConnection，BatchNormalization，BNN Activation を繰り返し実行し，最後に Argmax を実行する事で予測処理を行う．

文献 [6] に記載の予測プロトコルでは，行列積の代わりに XnorDotProduct が用いられているが，処理は行列積と変わらないため，Protocol 5 では行列積を行うように変更している．また，入力データに対しては XnorDotProduct と異なる処理をしているが，計算結果は行列積と変わらないため，同様に変更している．

**Protocol 5** Binarized Neural Networks

**Input:** $\boldsymbol{a}_0 \in \mathbb{Z}^{n_0}$：入力ベクトル
$\boldsymbol{W}_i \in \{1, -1\}^{n_{i-1} \times n_i}$：重み行列
$\boldsymbol{\theta}_i = (\boldsymbol{\gamma_i}, \boldsymbol{\mu_i}, \boldsymbol{\beta_i}, \boldsymbol{\sigma_i}) \in (\mathbb{R}^{n_i})^4$：BatchNormalization パラメータ
**Output:** $index \in \{1, \cdots, n_L\}$：予測結果
**Procedure:**
1: **for** $i = 1$ to $L - 1$ **do**
2: $\quad \boldsymbol{f}_i \leftarrow$FullConnection$(\boldsymbol{a}_{i-1}, \boldsymbol{W}_i, n_{i-1}, n_i)$
3: $\quad \boldsymbol{b}_i \leftarrow$BatchNormalization$(\boldsymbol{f}_i, \boldsymbol{\theta}_i, n_i)$
4: $\quad \boldsymbol{a}_i \leftarrow$BNN Activation$(\boldsymbol{b}_i, n_i)$
5: **end for**
6: $\boldsymbol{f}_L \leftarrow$FullConnection$(\boldsymbol{a}_{L-1}, \boldsymbol{W}_L, n_{i-1}, n_i)$
7: $\boldsymbol{b}_L \leftarrow$BatchNormalization$(\boldsymbol{f}_L, \boldsymbol{\theta}_L, n_i)$
8: $index \leftarrow$Argmax$(\boldsymbol{b}_L, n_i)$
9: Output $index$

BNN Activation プロトコルを Protocol 6 に示す．このプロトコルでは，ニューラルネットワークの Protocol 4 のように ReLU 関数を用いるのではなく，Sign 関数を用いる．これにより，中間ベクトル $\boldsymbol{a}_i$ の全ての要素を 1 または −1 にする．

**Protocol 6** BNN Activation

**Input:** $n \in \mathbb{Z}$：ベクトルサイズ
$\boldsymbol{b} \in \mathbb{R}^n$
**Output:** $\boldsymbol{a} \in \mathbb{R}^n$
**Procedure:**
1: **for** $j = 1$ to $n$ **do**
2: $\quad \boldsymbol{a}^{(j)} \leftarrow$Sign$(\boldsymbol{b}^{(j)})$
3: **end for**
4: Output $\boldsymbol{a}$

## 3. 秘密分散法とマルチパーティ計算

### 3.1 秘密分散法

秘密分散法は，秘密情報を $n$ 個の値に分散して管理する手法である．この分散された値をシェアと呼ぶ．特に $(k, n)$

閾値秘密分散法と呼ばれる手法は，シェアを $k$ 個以上集めることで秘密情報を復号することができるが，$k$ 個未満のシェアからは秘密に関する情報を一切得ることが出来ない．以降では，秘密分散法の一つである，Shamir$(k,n)$ 閾値秘密分散法に関して説明する．

Shamir$(k,n)$ 閾値秘密分散法 [9] は，以下のアルゴリズムにより分散と復号を行う．文中の $p$ は秘密分散法の法，$\ell$ は法 $p$ のビット数である．

- 分散
  $s$ を秘密情報とする．$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{k-1}x^{k-1} \ (mod\ p)$ $(\{a_1, \cdots, a_{k-1}\} \xleftarrow{\$} \mathbb{Z}_p^{k-1})$ を生成し，サーバ $i$ $(i = 1, 2, \cdots, n)$ に $[\![s]\!]_i = f(i)$ を送信する．
- 復号
  各サーバ $i$ は自分の持つシェア $[\![s]\!]_i$ を，他のサーバへ送信する．各サーバは，受け取った $(i, [\![s]\!]_i)$ を利用してラグランジュ補間公式（式 (1)）を用いて $f(0) = s$ を計算する．ここで $I$ は復号処理に参加するサーバの集合であり，$|I| \geq k$ ならば正しく復号ができる．

$$f(0) := \sum_{i \in I} [\![s]\!]_i \, \ell_i(0) \quad (I \subset \{1, \cdots, n\},\ |I| \geq k) \tag{1}$$

$$\ell_i(0) := \prod_{j \in I \setminus \{i\}} \frac{-j}{i-j}$$

### 3.2 マルチパーティ計算

マルチパーティ計算 (MPC) は，各参加者間で通信を行い，互いに自分の持つ秘密を漏らすことなく統計情報などを計算する手法のことである．秘密分散法では，シェアを持つ参加者間で MPC を行うことで，和や積，論理和，論理積を計算することが出来る．

前述の Shamir$(k,n)$ 閾値秘密分散法は，MPC により秘密を復号することなく和や積を計算することが出来る．等号判定や大小比較等の上位プロトコルは和と積の組み合わせで実現可能であるため，MPC により計算可能である [10][3].

Shamir$(k,n)$ 閾値秘密分散法での MPC では，シェア同士の和，シェアと平文の和，シェアと平文の積を求める計算は，他のサーバと通信することなく実行できるが，シェア同士の積を計算するにはサーバ間の通信が必要であるため，実行により多くの時間を要する．上位プロトコルは前述のとおり，和積の組み合わせで構成可能であるため，計算量の評価としてシェア同士の積プロトコルの実行回数を用いる．この実行回数のことを通信量と呼ぶ．また，並列に積プロトコルを実行する際は，その通信を一回にまとめることができる．そのようにして通信回数を削減した際の最小通信回数をラウンド数と呼ぶ．

### 3.3 表記

本稿で使用する記号やプロトコルの説明をする．

- $[\![a]\!]$ : $a$ のシェア．
- $[\![a]\!] + [\![b]\!]$ : $[\![a]\!]$ と $[\![b]\!]$ を入力とし，$[\![a+b]\!]$ を出力する和プロトコル．
- $[\![a]\!] + c$ : $[\![a]\!]$ と平文の $c$ を入力とし，$[\![a+c]\!]$ を出力する和プロトコル．
- $[\![a]\!] * [\![b]\!]$ : $[\![a]\!]$ と $[\![b]\!]$ を入力とし，$[\![a*b]\!]$ を出力する積プロトコル．
- $[\![a]\!] * c$ : $[\![a]\!]$ と平文の $c$ を入力とし，$[\![a*c]\!]$ を出力する積プロトコル．
- $[\![a < b]\!]$ : $[\![a]\!]$ と $[\![b]\!]$ を入力とし，$a$ と $b$ の大小を判定するプロトコル．$a < b$ の場合は $[\![1]\!]$ を返し，$a \geq b$ の場合は $[\![0]\!]$ を返す．
- $\left[\!\left[0 \leq b < \frac{p}{2}\right]\!\right]$ : $[\![a]\!]$ を入力とし，$a$ が 0 以上 $\frac{p}{2}$ 未満かを判定する区間判定プロトコル．$0 \leq b < \frac{p}{2}$ の場合は $[\![1]\!]$ を返し，そうでなければ $[\![0]\!]$ を返す．
- $[\![\bigwedge_{i=1}^{n} a_i]\!]$ : シェア $[\![a_i]\!]$ $(a_i \in \{0,1\})$ を入力とし，その全ての論理積 $\bigwedge_{i=1}^{n} a_i$ のシェアを出力するプロトコル（Unbounded Fan-In AND プロトコル）．
- Reveal$([\![a]\!])$ : $a$ を復号する．
- MatMult$([\![\boldsymbol{a}]\!], [\![\boldsymbol{W}]\!])$ : $\boldsymbol{a}$ と $\boldsymbol{W}$ の二つの行列のシェアを入力とし，行列積 $\boldsymbol{a} \times \boldsymbol{W}$ のシェアを出力するプロトコル．

行列積 MatMul に関しては，要素毎に積や和を計算しているだけなので，和プロトコルおよび積プロトコルを組み合わせて容易に計算できる．

大小判定プロトコル，区間判定プロトコルおよび Unbounded Fan-In AND プロトコルは，例えば文献 [10] で提案されている方式により計算できる．

## 4. システム

### 4.1 システム構成

本稿では，サービス提供者，$n$ 個の外部のクラウドサーバ，クライアントの三者が存在するモデルを考える．（**図 2**）

### 4.2 システムフロー

初めに，サービス提供者は平文の状態で学習処理を行い，予測モデルを作成する．次にサービス提供者は，予測モデルのうち，学習されたパラメータなど秘匿の必要性がある情報を秘密分散法により分散し，各サーバにシェアを配布する．また，全サーバに公開する必要のあるパラメータは平文のまま各サーバへ配布する．ユーザがシステムを利用する際，ユーザは入力データを秘密分散し，得られたシェアを各サーバに送信する各サーバは，サービス提供者から配布された予測モデルのシェアと，全サーバに公開されたパラメータ，及びユーザから受け取ったシェアを入力として MPC を実行することで，それぞれ予測結果のシェアを得る．最後に，各サーバはユーザに予測結果のシェアを送信する．ユーザは各サーバから送信されたシェアを復号す

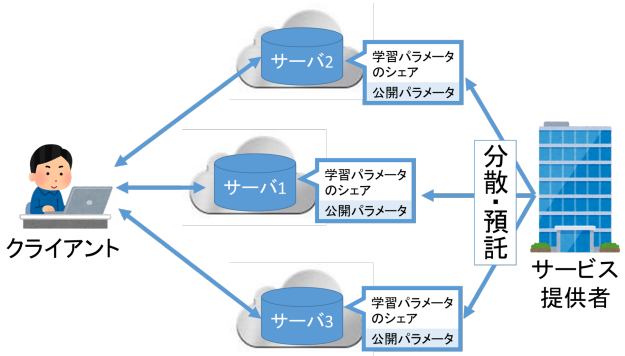


図 2　システム構成図

ることで，自身の入力に対応する予測結果を得る．

## 5. 提案手法

本章では BNN を用いた秘匿予測手法を提案する．

通常の Neural Networks は浮動小数点演算を繰り返し行っている．従来の秘匿予測手法では，計算量を減らすために，演算のたびに下位桁の切り捨てを行うなど，浮動小数点演算の精度を下げているため，予測の精度も低下していた．しかし，BNN は通常の Neural Networks とは異なり，各層の最後に Sign プロトコルを実行しているため，浮動小数点演算の精度をあまり下げることなく実行可能である．そこで本研究では，ニューラルネットワークとして BNN を適用する．

また，秘密分散法として Shamir$(k,n)$ 閾値秘密分散法を適用する．

### 5.1 予測モデルの分散

学習した重み行列 $\boldsymbol{W}_i$ $(i=1,\cdots,L-1)$ は Shamir$(k,n)$ 閾値秘密分散法を用いて各サーバに分散して保管する．ここで，ある $a$ を分散する際，$a \geq 0$ であれば $[\![a]\!]$ を，$a<0$ であれば $[\![p-a]\!]$ を分散することとする．そのため本稿で提案するシステムでは，$0 \leq a < \frac{p}{2}$ は正の数，$\frac{p}{2} \leq a < p$ は負の数を表す．

また重み行列と同じく学習した BatchNormalization パラメータに関しては，次の式 (2) および式 (3) の計算を事前に行い，$\boldsymbol{s}_i, \boldsymbol{t}_i$ $(i=1,2,\ \ldots\ ,L-1)$ を Shamir$(k,n)$ 閾値秘密分散法を用いて各サーバに分散して保管する．なお，$(L, n_0, \cdots, n_L)$ に関しては分散せず，平文のまま各サーバへ配布する．

$$\boldsymbol{s}_i \leftarrow \left\lfloor \frac{\boldsymbol{\gamma}_i}{\sqrt{\boldsymbol{\sigma}_i^2+\epsilon}} \times m \right\rfloor \tag{2}$$

$$\boldsymbol{t}_i \leftarrow \left\lfloor \left( \boldsymbol{\beta}_i - \frac{\boldsymbol{\mu}_i}{\sqrt{\boldsymbol{\sigma}_i^2+\epsilon}} \right) \times m \right\rfloor \tag{3}$$

本来の BNN では，BatchNormalization パラメータは浮動小数点数であったが，本提案ではある整数 $m$ を実験的に定め，$s_i$ と $t_i$ を $m$ 倍し，小数点以下を切り捨てることで整数とする．この $m$ をスケールパラメータとよぶ．

本章で提案する SecureBatchNormlization は線形計算であるため，秘密分散法の法 $p$ が十分に大きい場合，$m$ 倍するかしないかで正負が変わることはない．よって，SecureActivation 関数の出力 $a_i$ が変化することはなく，以降の計算には影響しない．

### 5.2 秘匿予測プロトコル

本節では，BNN を用いた秘匿予測プロトコル（SecureBNN）を提案する．SecureBNN で用いる大小判定，区間判定および Unbounded Fan-In AND の MPC プロトコルは，文献 [10] で提案されている方式を利用する．

SecureBNN プロトコルを Protocol 7 に示す．

**Protocol 7** SecureBNN

**Input:** $[\![\boldsymbol{a}_0]\!] \in \mathbb{Z}_p^{n_0}$：入力ベクトルのシェア
$[\![\boldsymbol{W}_i]\!] \in \{1,-1\}^{n_{i-1} \times n_i}$：重み行列のシェア $(i=1,\ \ldots\ ,L)$
$([\![\boldsymbol{s}_i]\!], [\![\boldsymbol{t}_i]\!])$：学習パラメータのシェア $(i=1,\ \ldots\ ,L)$
**Output:** $[\![index]\!] \in \{1,\cdots,n_L\}$：予測結果
**Procedure:**
1: **for** $i=1$ to $L-1$ **do**
2: $[\![\boldsymbol{f}_i]\!] \leftarrow$SecureFullConnection$([\![\boldsymbol{a}_{i-1}]\!], [\![\boldsymbol{W}_i]\!], n_{i-1}, n_i)$
3: $[\![\boldsymbol{b}_i]\!] \leftarrow$SecureBatchNormalization$([\![\boldsymbol{f}_i]\!], [\![\boldsymbol{s}_i]\!], [\![\boldsymbol{t}_i]\!], n_i)$
4: $[\![\boldsymbol{a}_i]\!] \leftarrow$SecureSign$([\![\boldsymbol{b}_i]\!], n_i)$
5: **end for**
6: $[\![\boldsymbol{f}_L]\!] \leftarrow$SecureFullConnection$([\![\boldsymbol{a}_{L-1}]\!], [\![\boldsymbol{W}_L]\!], n_{i-1}, n_i)$
7: $[\![\boldsymbol{b}_L]\!] \leftarrow$SecureBatchNormalization$([\![\boldsymbol{f}_L]\!], [\![\boldsymbol{s}_L]\!], [\![\boldsymbol{t}_L]\!], n_i)$
8: $[\![index]\!] \leftarrow$SecureArgmax$([\![\boldsymbol{b}_L]\!], n_i)$
9: Output $[\![index]\!]$

SecureFullConnection プロトコルを Protocol 8 に示す．このプロトコルでは，平文での FullConnection の計算と同じくシェア同士の行列積を計算する．

**Protocol 8** SecureFullConnection

**Input:** $n, m \in \mathbb{Z}$：ベクトルサイズ
$[\![\boldsymbol{a}]\!] \in \mathbb{Z}^n$：中間ベクトルのシェア，
$[\![\boldsymbol{W}]\!] \in \{1,-1\}^{n \times m}$：重み行列のシェア
**Output:** $[\![\boldsymbol{f}]\!] \in \mathbb{Z}_p^m$
**Procedure:**
1: $[\![\boldsymbol{f}]\!] \leftarrow$ MatMult $([\![\boldsymbol{a}]\!], [\![\boldsymbol{W}]\!])$；秘匿行列積
2: Output $[\![\boldsymbol{f}]\!]$

SecureBatchNormalization プロトコルを Protocol 9 に示す．平文での BatchNormalization では，平方根や除算といった計算が必要であったが，このプロトコルでは，式 (2) および式 (3) を事前計算することにより，加算と乗算のみで実行可能である．

SecureActivation プロトコルを Protocol 10 に示す．SecureActivation は，与えられた数値が正か 0 だった場合に

**Protocol 9** SecureBatchNormalization

**Input:** $n \in \mathbb{Z}$：ベクトルサイズ
$[\![\boldsymbol{f}]\!] \in \mathbb{Z}_p^n$：中間ベクトルのシェア，
$([\![\boldsymbol{s}]\!], [\![\boldsymbol{t}]\!])$ ←：学習パラメータのシェア
**Output:** $[\![\boldsymbol{b}]\!] \in \mathbb{Z}_p^n$
**Procedure:**
1: **for** $j = 1$ to $n$ **do**
2: $\quad [\![\boldsymbol{b}^{(j)}]\!] \leftarrow [\![\boldsymbol{s}^{(j)}]\!] \times [\![\boldsymbol{f}^{(j)}]\!] + [\![\boldsymbol{t}^{(j)}]\!]$
3: **end for**
4: Output $[\![\boldsymbol{b}]\!]$

は 1 を，負だった場合には −1 を返す．前提条件より，正の数は $(0 \le a < \frac{p}{2})$，負の数は $(\frac{p}{2} \le a < p)$ で表されるため，$\frac{p}{2}$ との大小判定を区間判定プロトコルにより計算する．

**Protocol 10** SecureActivation

**Input:** $n \in \mathbb{Z}$：ベクトルサイズ
$[\![\boldsymbol{b}]\!] \in \mathbb{Z}_p^n$：中間ベクトルのシェア
**Output:** $[\![\boldsymbol{a}]\!] \in \{1, -1\}^n$
1: **for** $j = 1$ to $n$ **do**
2: $\quad [\![\boldsymbol{a}^{(j)}]\!] \leftarrow 2 \times [\![0 \le \boldsymbol{b}^{(j)} < \frac{p}{2}]\!] - 1$
3: **end for**
4: Output $[\![\boldsymbol{a}]\!]$

SecureArgmax プロトコルを Protocol 11 に示す．初めに，全要素対の大小比較を計算する．この時，$\boldsymbol{b}^{(j)} \ge \boldsymbol{b}^{(k)}$ であれば $f^{(j,k)}$ には 1 が，$\boldsymbol{b}^{(j)} < \boldsymbol{b}^{(k)}$ であれば $f^{(j,k)}$ には 0 が格納される．

ベクトル $\boldsymbol{b}$ の中で最大の要素に注目すると，他の要素との比較結果はすべて 1 となり，最大値以外の要素に注目すると，少なくとも最大値の要素との比較結果が 0 となるため，比較結果の論理積を計算することで，最大値の要素のみを抽出することが可能である．

Argmax は，Protocol 11 のような総当り方式だけでなく，逐次最大値と比較する勝ち抜き方式や，トーナメント方式があるが，本稿では効率化のためラウンド数が最も少ない総当り方式を適用した．

**Protocol 11** SecureArgmax

**Input:** $n \in \mathbb{Z}$：ベクトルサイズ
$[\![\boldsymbol{b}]\!] \in \mathbb{Z}_p^n$：出力ベクトルのシェア
**Output:** $[\![index]\!] \in \{1, \cdots, n\}$：予測結果のシェア
**Procedure:**
1: **for** $j = 1$ to $n$ **do**
2: $\quad$ **for** $k = 1$ to $n$ **do**
3: $\quad\quad [\![f^{(j,k)}]\!] \leftarrow [\![\boldsymbol{b}^{(j)} \ge \boldsymbol{b}^{(k)}]\!]$
4: $\quad$ **end for**
5: **end for**
6: $[\![index]\!] \leftarrow 0$
7: **for** $j = 1$ to $n$ **do**
8: $\quad [\![index]\!] \leftarrow [\![index]\!] + [\![\wedge_{k=1}^{n} f^{(j,k)}]\!] * j$
9: **end for**
10: Output $[\![index]\!]$

## 6. 評価

本章では提案方式の評価を行う．評価に用いた環境を以下に示す．

**評価環境：**

- MNIST データセットを使用
- 入力データサイズ：8 *bit*
- 入力ノード数：$n_0 = 784$
- 中間ノード数：$n_1 = n_2 = 128$
- 出力ノード数：$n_3 = 10$

### 6.1 正当性評価

提案手法の正当性について評価する．提案手法と通常の BNN の処理との異なる点は，BatchNormalization パラメータを $m$ 倍し整数にしたことと，演算を $\mathbb{Z}_p$ 上で行うことの 2 点である．

BatchNormalization パラメータを整数に丸めた場合，従来の BNN よりも精度は落ちる．しかし，$m$ を大きくとることで影響を小さくし，精度を維持することができる．具体的に，本章冒頭に記載した評価環境の下で実験を行った結果，5.1 節で述べたスケールパラメータ $m$ を 100 にすることで，予測精度を 96.2% 程度にすることができた．

演算を $\mathbb{Z}_p$ 上で行う場合，前提条件より，値が $p/2$ を超えてしまうと正負が反転し，正しい計算結果を得ることができない．そこで $p$ を十分に大きくとる必要がある．本章冒頭に記載した評価環境において，SecureBNN がとりうる値の最大は，Protocol 7 のステップ 3 の $\boldsymbol{b}$ の要素がとりうる値の最大であり，最大値は $n_0 \times$ ( 入力データサイズ ) $\times$ ($s$ の取りうる最大 ) である．スケールパラメータ $m$ を 100 として実験を行った結果，$s$ の最大値は 429 となった．以上より，$n_0 \times$ ( 入力データサイズ ) $\times$ ($s$ の取りうる最大 ) $= 784 \times 2^8 \times 429 < 2^{27}$ であるため，$p$ として 28*bit* 以上の素数を選ぶことで，常に正しい計算結果を得ることができる．

### 6.2 安全性評価

本稿で用いている MPC プロトコルは semi-honest モデルにおいて安全であるため，提案手法において，ユーザからの入力データと予測モデルの情報は，サーバが semi-honest かつ閾値以上の結託がない限り漏えいすることはない．

重み行列や BatchNormalization パラメータは，秘密分散法により分散されて保管されているため，閾値以上のサーバが結託しない限り漏洩することはない．また計算途中の値に関する安全性は，用いている MPC プロトコルの安全性に帰着する．

### 6.3 計算量評価

3.2 節で述べた Shamir$(k, n)$ 閾値秘密分散法の評価手法

表 1　計算量評価

| | ラウンド数 | 通信量 |
|---|---|---|
| SecureFullConnection | 1 | $n_i$ |
| SecureBatchNormalization | 1 | $n_i$ |
| SecureActivation | 13 | $n_i(93\ell+1)$ |
| SecureArgmax | 16 | $\frac{1}{2}n_L(n_L-1)(279\ell+5)+5n_L^2$ |
| SecureBNN | 9L+3 | $(93\ell+3)\sum_{i=1}^{L-1}n_i+\frac{1}{2}n_L(n_L-1)(279\ell+5)+5n_L^2+2n_L$ |

である，通信量とラウンド数を用いてプロトコルごとに提案手法を評価し，表 1 に示す．

以下ではそれぞれのプロトコルに関して，詳細に評価する．

**SecureFullConnection**　Shamir 秘密分散法での行列積の通信量は出力行列の大きさに依存する．そのため次元数が $n_{i-1}$ のベクトル $a_{i-1}$，$n_{i-1}\times n_i$ の重み行列 $W_i$ を入力としたとき，SecureFullConnection の通信量は $n_i$ である．また，すべての積プロトコルを並列して実行することが出来るため，ラウンド数は 1 である．

**SecureBatchNormalization**　実行する積プロトコルは入力のベクトルの大きさに依存するため，通信量は $n_i$ である．また，すべての積プロトコルを並列して実行することが出来るため，ラウンド数は 1 である．

**SecureActivation**　SecureActivation では要素数分だけ区間判定プロトコルを実行する必要がある．文献 [10] より，区間プロトコルの通信量が $93\ell+1$，ラウンド数が 13 であるため，Sign プロトコル全体の通信量は $n_i(93\ell+1)$ である．また，全ての大小比較プロトコルを並列に実行できることから，ラウンド数は 13 である．

**SecureArgmax**　まず，全要素対の大小比較に関する計算量を評価する．大小比較プロトコルの通信量が $279\ell+5$，ラウンド数が 15 であるため，全体の通信量は $\frac{1}{2}n_L(n_L-1)(279\ell+5)$ であり，ラウンド数は 15 である．次に，大小比較の結果の AND を計算するために必要な計算量を評価する．ある配列の全ての AND を計算する，Unbounded Fan-in AND の通信量は，文献 [10] より $5n_L$，ラウンド数は 3 である．よって，全体の通信量は $5n_L^2$ であり，ラウンド数は 3 である．以上より，SecureArgmax 全体の通信量は $\frac{1}{2}n_L(n_L-1)(279\ell+5)+5n_L^2$ であり，ラウンド数は 16 である．

以上 4 つのプロトコルの通信量およびラウンド数をもとに，SecureBNN プロトコルの性能評価を行う．

それぞれのプロトコルを実行する回数は，Protocol 7 より，SecureFullConnection と SecureBatchNorm が $L$ 回，SecureActivation が $L-1$ 回，SecureArgmax が 1 回となっている．これより，通信量は $(93\ell+3)\sum_{i=1}^{L-1}n_i+\frac{1}{2}n_L(n_L-1)(279\ell+5)+5n_L^2+2n_L$ である．また，ラウンド数は $9L+3$ である．

## 7. 拡張検討

SecureActivation プロトコル（Protocol 10）に注目する．SecureActivation プロトコルは，SecureBatchNormalization プロトコルの出力が 0 以上であるかを判定している．ここで，$u\in\{1,-1\}$ を $s$ の符号部，$v\in\mathbb{N}$ を $s$ の整数部とする．この時，$sf+t$ の計算結果の符号と，$uf+\frac{t}{v}$ の計算結果の符号は同一になる．つまり，5.1 節で述べた $s,t$ と同等の方法で $\frac{t}{v}$ を事前計算することにより，SecureBNN プロトコル（Protocol 7）のステップ 3 の SecureBatchNormalization の処理は，$uf+\frac{t}{v}$ で置き換えることができる．

SecureBNN がとりうる値の最大が，Protocol 5 のステップ 3 の $b_1$ であることに注目すると，SecureBatchNormalization の処理を上記のように変更することにより，とりうる値の最大を，6.1 節で述べた数値よりも減らすことができると考えられる．

## 8. まとめ

本稿では，高い予測精度を達成可能で安全かつ効率的な秘匿化ニューラルネットワークを提案した．

提案方式では，ニューラルネットワークの一つである Binarized Neural Networks と秘密分散法を利用することで，入力データや学習パラメータを秘匿にしたまま，安全に予測処理を行う．

予測精度に関しては，秘密分散法とマルチパーティ計算により BNN プロトコルを再現しているため，従来手法に比べて予測精度の低下を抑えることができた．

**謝辞**　本研究の一部は，JST CREST JPMJCR1688 の支援を受けています．

### 参考文献

[1] M. Barni, C. Orlandi, and A. Piva. A privacy-preserving protocol for neural-network-based computation. In *Proceedings of the 8th Workshop on Multimedia and Security*, MM&Sec '06, pages 146–151, New York, NY, USA, 2006. ACM.

[2] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[3] Octavian Catrina and Sebastiaan de Hoogh. Improved

primitives for secure multiparty integer computation. In *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, pages 182–199, 2010.

[4] Herve Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. Cryptology ePrint Archive, Report 2017/035, 2017. `http://eprint.iacr.org/2017/035`.

[5] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 201–210. JMLR.org, 2016.

[6] M.Courbariaux, I.Hubara, D.Soudry, R.El-Yaniv, and Y.Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1., 2016. `https://arxiv.org/abs/1602.02830`.

[7] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 19–38, 2017.

[8] C. Orlandi, A. Piva, and M. Barni. Oblivious neural network computing via homomorphic encryption. *EURASIP J. Inf. Secur.*, 2007:18:1–18:10, January 2007.

[9] A. Shamir. How to share a secret. *In Communications of the ACM*, 22:612–613, 1979.

[10] K.Ohta T.Nishide. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. *Public Key Cryptography*, 207:343–360, 2007.