精度劣化を抑えた秘匿 BinarizedCNN の提案

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

概要：近年，機械学習手法の一種であるニューラルネットワークが画像認識などの様々な分野で活用されている．機械学習の普及に伴い，クラウドサービスを利用した機械学習サービスが一般的になってきた．しかしこういったサービスでは，ユーザの入力データや，サービス提供者が持つ学習モデルが情報漏えいする懸念がある．そこで本稿では，ユーザの入力と学習モデルを秘匿したまま，複雑なアーキテクチャで構成されたニューラルネットワークの予測処理を効率的に実行可能な秘匿 BinarizedCNN を提案する．本方式では，予測処理に必要な小数演算を等価の整数演算に変換し，秘密計算と相性の良いアーキテクチャを利用することで，CIFAR10 の予測タスクに対して実行時間を 15.05 秒，予測精度を 87.36% とした．

キーワード：秘密計算，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

Efficient Secure Binarized CNN Protocol Reducing Accuracy Degradation

Naohisa Nishida[1]　Tatsumi Oba[1]　Yuji Unagami[1]　Naoto Yanai[2]　Tadanori Teruya[3]
Nuttapong Attrapadung[3]　Takahiro Matsuda[3]　Goichiro Hanaoka[3]

Abstract: Recently, Neural Networks are used in various fields such as image recognition. With the spread of machine learning, machine learning services using cloud services have become popular. However, in such services, there is a concern that user input data and learning models possessed by service providers may leak information. In this paper, we propose Secure Binarized CNN that can efficiently compute Neural Network prediction processing while preserving the privacy of user input and learning model. We proposed a method for converting decimal arithmetic to equivalent integer arithmetic, and an architecture that is compatible with secret computation. As a result, the execution time was set to 15.05 seconds and the prediction accuracy was set to 87.36% with CIFAR10 dataset.

Keywords: Secure Computation, Secret Sharing, Multi-Party Computation, Machine Learning, Neural Networks

1. はじめに

1.1 背景

近年，機械学習手法の一種であるニューラルネットワークが画像認識や自然言語処理などの様々な分野で活用されている．ニューラルネットワークは，学習フェーズと予測フェーズの 2 つのフェーズで成り立っている．学習フェーズでは，学習データと呼ばれる既知のデータから学習モデルを作成し，予測フェーズでは，サービスを受けるユーザが自身のデータを入力し，学習データを用いて予測結果を計算する．

ニューラルネットワークの普及や実用性の向上に伴い，MLaaS(Machine Learning as a Service) と呼ばれる学習モデルを利用したクラウドサービスの利用が一般的になってきた．しかし，予測フェーズの入力情報はユーザのプライバシー情報を含むことも多いため，サービス提供者やク

[1] パナソニック株式会社
Panasonic Corporation
[2] 大阪大学大学院　情報科学研究科
Graduate School of Information Science and Technology, Osaka University
[3] 国立研究開発法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

© 2019 Information Processing Society of Japan

ラウドサーバに対して秘匿されることが望ましい．また，学習モデルの作成に用いる学習データは，対価を支払って得たデータであることが多く，加えて学習モデルの作成には莫大な時間を要する．そのため，学習モデルはサービス提供者にとって価値のある資産であり，ユーザやクラウドサーバに対して秘匿されることが望ましい．

この問題を解決するため，学習モデルとユーザの入力情報を秘匿したままニューラルネットワークを実行する方式が提案されている [1], [4], [5], [6], [9]．しかしこれらの多くの既存研究では，CIFAR10 データセットを用いた物体識別など，多くの計算量を必要する複雑なアーキテクチャを用いたニューラルネットワークに対してはアプローチを行っていない．

CIFAR10 で用いる複雑なアーキテクチャの一例として Convolutional Neural Networks（CNN）が挙げられる．CNN は通常のニューラルネットワークに比べて高い予測精度を出すことが可能である反面，複雑な処理が必要になる．一般にニューラルネットワークを秘密計算で実行する場合，秘密計算化に伴う計算誤差が大きくなり，平文での実行に比べて精度が劣化する．CNN を秘密計算化すると，通常のニューラルネットワークに比べて精度が大きく劣化してしまうことが見込まれる．また，精度の劣化を抑えようとすると計算量が膨大になるという問題もある．

そこで本稿では，学習モデルとユーザの入力を秘匿したまま実行可能なニューラルネットワークに関して，計算量を抑えつつ，精度の劣化も抑える方式を提案する．また，秘密計算と相性がよく精度の高いアーキテクチャを探索し，より高精度な方式の提案を目指す．

1.2 想定シナリオ

本稿で想定するシナリオについて説明する．図 1 は想定するシナリオを表した図であり，学習モデルを作成する病院（サービス提供者），秘匿機械学習を実行するクラウドサーバ，患者情報を入力として学習処理を依頼する医師（ユーザ）を想定する．

病院は，自身の所持している学習データを利用して学習モデルを作成し，秘匿化して機械学習アーキテクチャと共にクラウドサーバに送信する．医師は患者の情報を秘匿化してクラウドサーバに送信し，機械学習による病名予測などを依頼する．クラウドサーバは秘匿化された学習モデルと患者情報を基に秘匿機械学習処理を行い，秘匿された結果を医師に返す．こうすることで，患者情報は医師以外には秘匿され，学習モデルは病院以外に秘匿される．

1.3 関連研究

近年，ユーザの入力情報を秘匿したままニューラルネットワークを実行する様々な方式 [1], [4], [5], [6], [9] が提案されている．

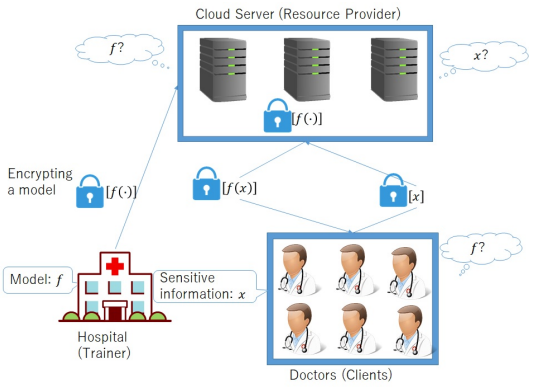

図 1　想定シナリオ

Gazelle[4]，XONN[6] は学習モデルを秘匿できないが，CIFAR10 を対象に大きなアーキテクチャを用いた実験を行っている．SecureML[5]，SecureQNN[1], SecureNN[9] は学習モデルも秘匿可能な方式である．しかし，いずれも MNIST のみを対象としており，層の少ないアーキテクチャのみを対象にしている．

XONN[6] については，本稿で説明するアプローチと同様の方式を提案しているが，BatchNormalization パラメータが正であることが仮定されているため，適用可能なアーキテクチャが限られてしまう．

2. ニューラルネットワーク

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，学習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学習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．

通常のニューラルネットワークでは，FullConnection，BatchNormalization，Activation と呼ばれる処理の組み合わせで構成されることが多い．この構成でも，MNIST データセットを用いた手書き数字の識別のような問題を高精度に解くことが可能であるが，より複雑な予測を行うためには後述する CNN 等のニューラルネットワークの改良方式を用いることが望ましい．

以下に FullConnection，BatchNormalization，Activation の各プロトコルを示す．

FullConnection のプロトコルを Protocol 1 に示す．このプロトコルは，入力ベクトル $\boldsymbol{X}$ と学習した重み行列 $\boldsymbol{W}$ に対し，行列積 $\boldsymbol{X} \times \boldsymbol{W}$ を計算する．

Protocol 1 FullConnection

Input: $h, w \in \mathbb{Z}$：行列サイズ
$\boldsymbol{X} \in \mathbb{R}^h$：入力ベクトル
$\boldsymbol{W} \in \mathbb{R}^{h \times w}$：重み行列
Output: $\boldsymbol{X'} \in \mathbb{Z}^w$
Procedure:
1: $\boldsymbol{X'} \leftarrow \boldsymbol{X} \times \boldsymbol{W}$；行列積

BatchNormalization のプロトコルを Protocol 2 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 2 BatchNormalization

Input: $h \in \mathbb{Z}$：行列サイズ
$\boldsymbol{X} \in \mathbb{R}^h$：入力ベクトル
$\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\mu}, \boldsymbol{\beta}, \boldsymbol{\sigma}) \in \left(\mathbb{R}^h\right)^4$：BatchNormalization パラメータ
ϵ：小さい正の定数
Output: $\boldsymbol{X'} \in \mathbb{R}^h$
Procedure:
1: **for** $i = 1$ to h **do**
2: $\boldsymbol{X}'_i \leftarrow \frac{\boldsymbol{X}_i \boldsymbol{\gamma}_i}{\sqrt{\boldsymbol{\sigma}_i{}^2 + \epsilon}} + \boldsymbol{\beta}_i - \frac{\boldsymbol{\mu}_i}{\sqrt{\boldsymbol{\sigma}_i{}^2 + \epsilon}}$
3: **end for**

Activation のプロトコルを Protocol 3 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 3 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 3 Activation

Input: $n \in \mathbb{Z}$：ベクトルサイズ
$\boldsymbol{X} \in \mathbb{R}^n$：入力ベクトル
Output: $\boldsymbol{X'} \in \mathbb{R}^n$
Procedure:
1: **for** $i = 1$ to n **do**
2: $\boldsymbol{X}'_i \leftarrow \mathrm{ReLU}(\boldsymbol{X}_i)$
3: **end for**

2.1 Convolutional Neural Networks(CNN)

CNN は，Convolution と呼ばれる処理を含むニューラルネットワークの総称である．一般に，FullConnection，BatchNormalization，Activation のみで構成されたニューラルネットワークと比べて，CNN は精度が向上する傾向にある．しかしその一方で，CNN は計算量が膨大となるため，計算量を削減する工夫が求められる．

以下に Convolution プロトコルの一例と，CNN で用いられる BatchNormalization3D プロトコル，CNN で用いられることの多い Pooling プロトコルを示す．

Convolution のプロトコルを Protocol 4 に示す．このプロトコルは，入力テンソル $\boldsymbol{X}$ と学習したフィルタ $\boldsymbol{W}$ に対し，畳み込みを計算する．行列積 $\boldsymbol{a} \times \boldsymbol{W}$ を計算する．

Protocol 4 Convolution

Input: $h, w, d \in \mathbb{Z}$：テンソルサイズ
$\boldsymbol{X} \in \mathbb{R}^{h \times w \times d}$：入力データ
$n, m \in \mathbb{Z}$：フィルタサイズ
$\boldsymbol{W} \in \mathbb{R}^{n \times m \times m \times d}$：フィルタ
Output: $\boldsymbol{X'} \in \mathbb{R}^{h \times w \times n}$：$\boldsymbol{X}$ と $\boldsymbol{W}$ の畳み込み
Procedure:
1: **for** $i = 1$ to $h - m - 1$ **do**
2: **for** $j = 1$ to $w - m - 1$ **do**
3: **for** $k = 1$ to n **do**
4: $\boldsymbol{T} \leftarrow \boldsymbol{X}_{i \cdots i+m, j \cdots j+m, 1 \cdots d} \in \mathbb{R}^{m \times m \times d}$ を抽出
5: $\boldsymbol{X}'_{i,j,k} \leftarrow \boldsymbol{T}$ と $\boldsymbol{W}_k \in \mathbb{R}^{m \times m \times d}$ を畳み込み
6: **end for**
7: **end for**
8: **end for**

BatchNormalization3D のプロトコルを Protocol 5 に示す．このプロトコルは，BatchNormalization プロトコルの入力が 3 次元テンソルになったものである．

Protocol 5 BatchNormalization3D

Input: $h, w, d \in \mathbb{Z}$：テンソルサイズ
$\boldsymbol{X} \in \mathbb{R}^{h \times w \times d}$：入力テンソル
$\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\mu}, \boldsymbol{\beta}, \boldsymbol{\sigma}) \in \left(\mathbb{R}^d\right)^4$：BatchNormalization パラメータ
ϵ：小さい正の定数
Output: $\boldsymbol{X'} \in \mathbb{R}^{h \times w \times d}$
Procedure:
1: **for** $i = 1$ to h **do**
2: **for** $j = 1$ to w **do**
3: **for** $k = 1$ to d **do**
4: $\boldsymbol{X}'_{i,j,k} \leftarrow \frac{\boldsymbol{X}_{i,j,k} \boldsymbol{\gamma}_k}{\sqrt{\boldsymbol{\sigma}_k{}^2 + \epsilon}} + \boldsymbol{\beta}_k - \frac{\boldsymbol{\mu}_k}{\sqrt{\boldsymbol{\sigma}_k{}^2 + \epsilon}}$
5: **end for**
6: **end for**
7: **end for**

Pooling のプロトコルを Protocol 6，Protocol 7 に示す．Pooling には図 2 のように MaxPooling や AveragePooling，SumPooling などが用いられる．一般的に MaxPooling が最も精度がよくなるが，AveragePooling や SumPooling 等に比べて必要な計算量が多くなる．

Protocol 6 Pooling(MaxPooling)

Input: $h, w, d \in \mathbb{Z}$：テンソルサイズ
$\boldsymbol{X} \in \mathbb{R}^{h \times w \times d}$：入力テンソル
$n \in \mathbb{Z}$：フィルタサイズ
Output: $\boldsymbol{X'} \in \mathbb{R}^{h/n \times w/n \times d}$
Procedure:
1: **for** $k = 1$ to d **do**
2: **for** $i = 1$ to h/n step n **do**
3: **for** $j = 1$ to w/n step n **do**
4: $\boldsymbol{X}'_{i,j,k} \leftarrow max(\boldsymbol{X}_{(i-1)*n+1 \cdots i*n, (j-1)*n+1 \cdots j*n, k})$
5: **end for**
6: **end for**
7: **end for**

Protocol 7 Pooling(SumPooling)

Input: $h, w, d \in \mathbb{Z}$：テンソルサイズ
$\boldsymbol{X} \in \mathbb{R}^{h \times w \times d}$：入力テンソル
$n \in \mathbb{Z}$：フィルタサイズ
Output: $\boldsymbol{X'} \in \mathbb{R}^{h/n \times w/n \times d}$
Procedure:
1: **for** $k = 1$ to d **do**
2: **for** $i = 1$ to h/n step n **do**
3: **for** $j = 1$ to w/n step n **do**
4: $\boldsymbol{X'}_{i,j,k} \leftarrow \Sigma(\boldsymbol{X}_{(i-1)*n+1 \cdots i*n,(j-1)*n+1 \cdots j*n,k})$
5: **end for**
6: **end for**
7: **end for**

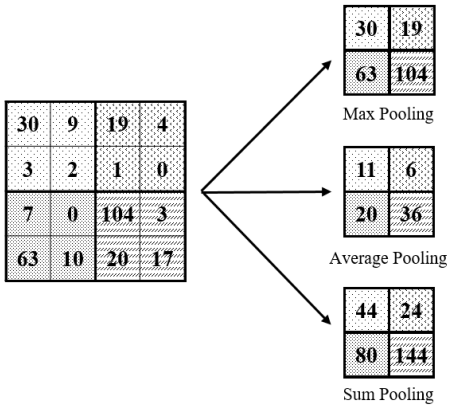

図 2　Pooling 手法

2.2 Binarized Neural Networks(BNN)

BNN[3] は，FullConnection における重み行列や，Convolution におけるフィルタの要素が $\{-1, 1\}$ の二値で構成され，予測時の Activation に用いられる関数が Sign 関数（Protocol 8）であるようなニューラルネットワークである．値が二値化されるため処理が簡単になり計算量が削減されるが，精度は大きく劣化する．精度の劣化に関しては，層の深いアーキテクチャを用いることで，通常のニューラルネットワークと同等の精度を達成することが可能となる．

また BNN は，BatchNormalization 以外の全てのパラメータが整数で表されているため，計算量の多い小数演算を実行することなく秘密計算で実行することが出来，かつ秘密計算化に伴ってパラメータを整数もしくは固定小数点数に変換しなくてもよいので，精度の劣化が少ない．

本稿では Convolution を含む BNN を明示的に BinarizedCNN と表す．

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 (t, n) 閾値秘密分散法と呼ばれる手法は，シェアを t 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，t 個未満の

Protocol 8 Sign

Input: $h \in \mathbb{Z}$：ベクトルサイズ
$\boldsymbol{X} \in \mathbb{R}^h$：入力ベクトル
Output: $\boldsymbol{X'} \in \mathbb{R}^h$
Procedure:
1: **for** $i = 1$ to h **do**
2: $\boldsymbol{X}'_i \leftarrow \begin{cases} 1 & \boldsymbol{X}_i \geq 0 \\ -1 & \boldsymbol{X}_i < 0 \end{cases}$
3: **end for**

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(t, n) 閾値秘密分散法に関して説明する．

Shamir(t, n) 閾値秘密分散法 [7] 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，ℓ は法 p のビット数である．

- 分散
  s を秘密情報とする．$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \ (mod\ p)$ $(\{a_1, \cdots, a_{k-1}\} \xleftarrow{\$} \mathbb{Z}_p^{t-1})$ 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，$|I| \geq t$ ならば正しく復号ができる．

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(t, 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 [8][2].

Shamir(t, 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回数を用いる．この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 1 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．

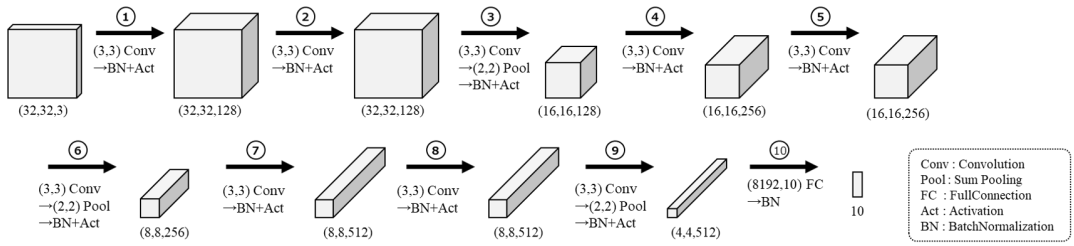

図 3 BNN Architecture for CIFAR-10.

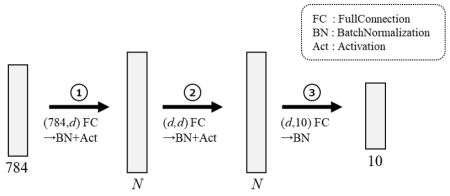

図 4 BNN Architecture for MNIST

4. 課題とアプローチ

4.1 課題

既存研究として，学習モデルとユーザの入力情報を秘匿したままニューラルネットワークを実行する方式が提案されている [1], [4], [5], [6], [9]．しかしこれらの多くの既存研究では，CIFAR10 データセットを用いた物体識別など，多くの計算を必要する複雑なアーキテクチャを用いたニューラルネットワークに対してはアプローチを行っていない．

通常，ニューラルネットワークの処理は小数演算が必要であるため，秘密計算でこれらを実行するためには，処理の重い小数演算をそのまま行うか，パラメータを整数化して計算を近似，簡略化するしかない．多くの計算を必要する複雑なアーキテクチャを対象にした場合，小数演算をそのまま行うと計算量が膨大になり，整数化すると計算誤差が発生し予測精度が劣化してしまう．

これらの問題を解決するため，次節に述べるアプローチを提案する．

4.2 BatchNormed-Activation

2.2 節で述べたように，Convolution，FullConnection のパラメータが二値化されてるため，BNN の処理のうち BatchNormalization 以外の処理は全て整数演算のみで実行可能である．そのため，BatchNormalization の処理を，平文での処理と誤差なく実行することが出来れば，秘密計算化に伴う精度の劣化をなくすことが出来，精度を高いまま維持することが出来る．そこで本節では，BatchNormalization と Activation をまとめて実行することで，計算誤差，精度劣化をなくす方式について述べる．

BatchNormalization と Activation は式 1 で表現できる．

$$X'_{i,j,k} \leftarrow Sign\left(\frac{\boldsymbol{X}_{i,j,k}\gamma_k}{\sqrt{{\sigma_k}^2+\epsilon}} + \beta_k - \frac{\mu_k}{\sqrt{{\sigma_k}^2+\epsilon}}\right) \tag{1}$$

これを変形し，Sign 関数を展開すると式 2 と表現できる．

$$X'_{i,j,k} \leftarrow \begin{cases} 1 & \boldsymbol{s}_k * \boldsymbol{X}_{i,j,k} + \boldsymbol{t}_k \geq 0 \\ -1 & otherwise \end{cases} \tag{2}$$

$$\boldsymbol{s}_k = \frac{\gamma_k}{\sqrt{{\sigma_k}^2+\epsilon}} \boldsymbol{t}_k = \beta_k - \frac{\mu_k}{\sqrt{{\sigma_k}^2+\epsilon}}$$

式 2 の場合分け条件の不等号の両辺を $\boldsymbol{s}_k$ で割ることで，式 3 のように変換できる．

$$X'_{i,j,k} \leftarrow \begin{cases} 1 & (\boldsymbol{s}_k > 0 \text{ かつ } \boldsymbol{X}_{i,j,k} + \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor \geq 0) \\ & \text{または } (\boldsymbol{s}_k < 0 \text{ かつ } \boldsymbol{X}_{i,j,k} + \left\lceil \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rceil \leq 0) \\ -1 & otherwise \end{cases} \tag{3}$$

また，秘密分散法のように，全ての計算を法 p 上で計算する場合，式 3 は式 4 のように変換できる．なお，$0 \leq a \leq \frac{p-1}{2}$ を正の値，$\frac{p+1}{2} \leq a \leq p-1$ を負の値として扱うものとする．

$$X'_{i,j,k} \leftarrow \begin{cases} 1 & (\boldsymbol{s}_k > 0 \text{ かつ } \boldsymbol{X}_{i,j,k} + \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor \geq 0) \\ & \text{または } (\boldsymbol{s}_k < 0 \text{ かつ } \boldsymbol{X}_{i,j,k} + \left\lceil \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rceil + \frac{p-1}{2} \geq 0) \\ -1 & otherwise \end{cases} \tag{4}$$

式 3 の 2 つ目の条件式は，法 p 上での計算の場合，$\frac{p+1}{2} \leq \boldsymbol{X}_{i,j,k} + \boldsymbol{t}_k \leq p$ と言い換えることが出来る．そのため，式 4 のように $\frac{p-1}{2}$ を加えることで，2 つ目の条件式は $0 \leq \boldsymbol{X}_{i,j,k} + \boldsymbol{t}_k \leq \frac{p-1}{2}$ となり，$\boldsymbol{X}_{i,j,k} + \boldsymbol{t}_k \geq 0$ と書き換えることが出来る．

本稿で提案するシステムは，学習は平文で行うため，事前に $\boldsymbol{s}_k$ が 0 以下であるかを判断可能である．そこで，式 5 を事前に計算しておくことで，式 6 と変換することが出来る．

$$\boldsymbol{v}_k \leftarrow \begin{cases} \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor & \boldsymbol{s}_k > 0 \\ \left\lceil \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rceil + \frac{p-1}{2} & \boldsymbol{s}_k < 0 \end{cases} \tag{5}$$

$$X'_{i,j,k} \leftarrow \begin{cases} 1 & \boldsymbol{X}_{i,j,k} + \boldsymbol{v}_k \geq 0 \\ -1 & otherwise \end{cases} \quad (6)$$

続けて，式 5 における $\boldsymbol{v}_k$ の値が $\boldsymbol{X}$ のとり得る値の範囲に比べて大きい，または小さい場合について考える．まず，本システムへの入力値の範囲と学習モデルが分かっている状態であれば，各層での計算途中の値や出力の値（つまり，次の層の入力の値）がとりうる範囲を計算することが出来る．そこで，BatchNormalizaion への入力値の範囲が $\{-u, u\}$ であると仮定する．この時，$\boldsymbol{v}_k$ が u 以上であれば Activation の出力は必ず 1 となり，$-u$ よりも小さいければ必ず -1 となる．また，$\boldsymbol{s}_k = 0$ の場合は，Activation の出力は $\boldsymbol{t}_k$ が 0 以上であるかに依存する．これらを加味し，$\boldsymbol{v}_k$ を式 7 のように計算することで，精度劣化なく秘密計算化が可能である．

$$\boldsymbol{v}_k \leftarrow \begin{cases} u & (s = 0 \text{ かつ } \boldsymbol{t}_k \geq 0) \\ & \text{または } (\boldsymbol{s}_k > 0 \text{ かつ } \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor \geq u) \\ & \text{または } (\boldsymbol{s}_k < 0 \text{ かつ } \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor \leq -u) \\ -u - 1 & (s = 0 \text{ かつ } \boldsymbol{t}_k < 0) \\ & \text{または } (\boldsymbol{s}_k > 0 \text{ かつ } \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor < -u) \\ & \text{または } (\boldsymbol{s}_k < 0 \text{ かつ } \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor > u) \\ \left\lfloor \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rfloor & \boldsymbol{s}_k > 0 \\ \left\lceil \frac{\boldsymbol{t}_k}{\boldsymbol{s}_k} \right\rceil + \frac{p-1}{2} & \boldsymbol{s}_k < 0 \end{cases} \quad (7)$$

上で述べた説明は BatchNormalization の直後に Sign が実行されるアーキテクチャを対象としているが，BatchNormalization と Activation の間に MaxPooling，AveragePooling，SumPooling のいずれかが実行されるようなアーキテクチャに対しても適用可能である．

BatchNormalization と Activation の間に MaxPooling が実行されるようなアーキテクチャの場合，MaxPooling を Sign の後に実行しても結果は変わらないことから，アーキテクチャを BatchNormalization，Activation，MaxPooling の順に変更することで，本アプローチを適用可能である．

BatchNormalization と Activation の間に AveragePooling，または SumPooling が実行されるようなアーキテクチャの場合，AveragePooling または SumPooling を BatchNormalization の直前に実行しても結果は変わらない．そのため，アーキテクチャを AveragePooling(SumPooling)，BatchNormalization，Activation の順に変更することで，本アプローチを適用可能である．

4.3 アーキテクチャの探索

多くの既存研究では既存のアーキテクチャを用いて実験を行っている．しかし，秘密計算と相性の良いアーキテクチャを用いることで，実用的な実行時間を維持したまま高い精度を維持できる可能性がある．そこで，BNN 論文に記載のアーキテクチャを改良した，いくつかの新しいアーキテクチャで実験を行い，精度評価を行った．

まず，含めることで精度が向上する可能性のある Pooling と呼ばれる処理に関して考察を行った．Pooling のうち，MaxPooling と呼ばれる処理を含んだアーキテクチャの多くは，含んでいないものに比べて精度が向上する傾向にある．しかし，MaxPooling は出力するテンソルサイズに応じた回数の大小比較を必要とするため，計算量が膨大になってしまう．そこで MaxPooling の代用として，MaxPooling ほどではないが精度が向上する SumPooling を用いる．

また，SumPooling 層の間に実行する Convolution 層，BatchNormalization 層，Activation 層を増やすことでも精度向上が見込まれるため，増加する計算量も考慮し，層の数，Convolution フィルタサイズを変化させて実験し，決定する．

本稿では，CIFAR10 向けアーキテクチャとして図 3 を，MNIST 向けアーキテクチャとして図 4 を採用した．

5. 提案システム

本章では，前述のアプローチを用いた秘匿 BinarizedCNN を提案する．

5.1 システムフロー

以下に，サービス提供者が事前に実行するフローを示す．

(1) 4.3 節で提案したアプローチにしたがってアーキテクチャを決定する
(2) 平文の状態で学習処理を行い，学習モデルを作成する
(3) 4.2 節で述べた方法で BatchNormalization パラメータを変換する
(4) 後に Activation が実行されない BatchNormalization のパラメータは文献 [10] の方法で変換する
そのために，変換に必要なスケールパラメータ q を実験的に決定する
(5) 法 p を決定する
(6) 学習モデルなど，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
(7) 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

以下に，ユーザがサービスを利用する際のフローを示す．

(1) 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
(2) 各サーバは，サービス提供者から配布された学習モデルのシェアと，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入力データのシェアを入力として MPC を実行し，予測結果のシェアを得る
(3) 各サーバはユーザに予測結果のシェアを送信する
(4) ユーザは各サーバから送信されたシェアを復号するこ

表 1　計算量

	ラウンド	通信量
秘匿 BinarizedCNN（CIFAR-10）	80	$696330 + 516096 \times (93\ell + 1)$
SecureConvolution	1	$h * w * d \times c'$
SecureBatch-normalizedActivation	13	$h * w * d \times c \times (93\ell + 1)$
SecureSumPooling	0	0
SecureFullConnection	1	h
SecureBatchNormalization	1	$h * w * d$
秘匿 BNN（MNIST）（N=128）	24	$1050 + 256 \times (93\ell + 1)$
秘匿 BNN（MNIST）（N=1000）	24	$2794 + 2000 \times (93\ell + 1)$

とで，自身の入力に対応する予測結果を得る

5.2 スケールパラメータ q の決定

文献 [10] で提案されている BatchNormalizaion 変換手法で用いるスケールパラメータ q を決定する．スケールパラメータ q は，値を大きくすれば精度劣化が少なくなるが，計算途中の数値や計算結果の数値が大きくなり，必要な法 p のビット数が増えてしまう．秘密分散ベースのマルチパーティプロトコルの計算量は p のビット数に依存することが多いため，法 p のビット数が増えてしまうと計算量が膨大になってしまう．そこで，学習時に q の値を変えて実験を行い，適切な q を設定する．

5.3 法 p の決定

本提案システムは，入力データの値域と学習モデルが分かっている状態であれば，Convolution 層や FullConnection 層などの全層の計算途中の数値や計算結果の数値の範囲を求めることが出来る．例えばある Convolution 層での入力データの値の絶対値の上界が X で，Convolution フィルタの要素の絶対値の上界が C，フィルタサイズが $h * w * d$ である場合，この Convolution 層で現れる数値の範囲は $\{-h * w * d * X * C, h * w * d * X * C\}$ となる．これを全ての層に対して計算することで，システム全体で現れる数値の範囲 $\{-u, u\}$ を求められる．法 p はこの数値 u を用いて，$2 * u$ 以上の素数を選べばよい．

5.4 ラウンド数と通信量

以下に，提案した各秘匿予測プロトコル及び，システム全体の通信量とラウンド数を記載する．また本節の通信量とラウンド数は，提案プロトコル/システムを文献 [8] に記載の MPC プロトコルで実現した場合の数値である．表中の ℓ は法 p のビット数であり，h, w, d は各関数の出力テンソルのサイズを示す．

6. 実験

提案システムを実装し，実行時間，通信データサイズ及び予測精度の評価を行った．既存研究との比較のため，CIFAR10 向けアーキテクチャだけでなく，MNIST 向けアーキテクチャでの実験も行った．

6.1 実験環境

実験では，本提案システムを Shamir(2,3) 閾値秘密分散ベースの MPC を用いて実装した．実装には C++, g++6.4.1 を用いた．実験では Amazon EC2 の c4.8xlarge インスタンスを用いた．同一リージョン内に 3 台のインスタンスを立ち上げることで，LAN 環境を疑似的に再現した．

MPC プロトコルには文献 [2] と文献 [8] に記載の方式がよく知られているが，文献 [2] の方式は大きな法 p を用いなければならず通信量が膨大になるため，本稿では文献 [8] の方式を用いることとした．秘密分散には Shamir(t, n) 秘密分散を採用し，閾値 $\boldsymbol{t = 2}$，パーティ数 $\boldsymbol{n = 3}$ として実験を行った．

予測精度に関しては，平文で同等の処理を実装して評価を行った．通信データサイズは実際に通信したデータサイズを計測し，記載した．実行時間については，乱数生成などの入力に依存せず，事前に計算できる処理にかかった時間を Offline 時間として，入力に依存する処理にかかった時間を Online 時間としてそれぞれ計測した．特に Online 時間は，実際のシステムを運用する際に重要になるため，Online 時間が少ない方が望ましい．

6.2 設定

6.2.1 CIFAR10

CIFAR10 データセットを想定した実験では，図 3 に示すアーキテクチャを用いる．実験的に求めたスケールパラメータ q を $\boldsymbol{q = 600}$ と設定し，適切な法 p として $\boldsymbol{p = 2^{16} - 17}$ を用いる．

6.2.2 MNIST

MNIST データセットを想定した実験では，図 4 に示すアーキテクチャを用いる．図中の中間層の値である N は $\boldsymbol{N = 128}$ と $\boldsymbol{N = 1000}$ にした場合の 2 つについて実験を行った．それぞれ，$\boldsymbol{N = 128}$ とした場合については，スケールパラメータ q を $\boldsymbol{q = 10000}$，法 p を $\boldsymbol{p = 2^{19} - 1}$ とし，$\boldsymbol{N = 1000}$ とした場合については，スケールパラメータ q を $\boldsymbol{q = 10000}$，法 p を $\boldsymbol{p = 2^{20} - 5}$ とした．

表 2　CIFAR-10 Result

	モデル秘匿	秘匿学習	Time (sec)			Comm. size (MB)		予測精度 (%)
			Offline	Online	合計	Offline	Online	
Gazelle [4]			9.34	3.56	12.9	940	296	81.61
XONN [6]			—	—	5.79	—	—	81.85
提案手法	✓		12.38	2.67	15.05	410	58.4	87.36

表 3　MNIST Result

	モデル秘匿	秘匿学習	Time (sec)			Comm. size (MB)		予測精度 (%)
			Offline	Online	合計	Offline	Online	
SecureML [5]	✓	✓	4.7	0.18	4.88	—	—	93.1
Gazelle [4]			0	0.03	0.03	0	0.5	97.6
XONN [6]			—	—	0.13	—	—	97.6
SecureNN [9]	✓	✓	0.481	0.33	0.81	47.5	22.5	98.77
SecureQNN [1]	✓		0	0.02	0.02	—	2.41	93.4
SecureQNN2 [1]	✓		0	0.08	0.08	—	33.12	99.0
提案手法 ($d = 128$)	✓		0.04	0.01	0.05	0.08	1.27	95.9
提案手法 ($d = 1000$)	✓		0.35	0.03	0.38	0.62	13.4	97.94

6.3　結果

実験結果を表 2 および表 3 に示す．

CIFAR10 を対象とした実験に関しては，精度が 87.36%，Offline 時間が 12.38 秒，Online 時間が 2.67 秒となった．この性能は，モデルを秘匿できていない Gazelle と比べて，Online 時間は 0.89 秒，精度は 5.75% 上回っている．また，モデルを秘匿できてない XONN と比較しても，実行時間に関しては正確に比較できないが，実行時間を大きく増やすことなく予測精度を 5.56% 向上出来ている．

MNIST を対象として実験に関しては，$N = 128$ の場合について，精度が 95.9%，Offline 時間が 0.04 秒，Online 時間が 0.01 秒となった．また，$N = 1000$ とした場合，精度が 97.4%，Offline 時間が 0.35 秒，Online 時間が 0.03 秒となった．この性能は，Online 時間と予測精度に着目すると，モデルを秘匿できていない Gazelle[4] や XONN[6] 等と比べて同程度の性能であることがわかる．また，モデルを秘匿できている SecureQNN[1] と比べた場合，Online 時間，予測精度共に上回っている．SecureNN[9] や SecureQNN のもう一つの実装と比較すると，予測精度は 1% 程度下回っているが，Online 時間は上回る結果となった．

7.　まとめ

本稿では，学習モデルとユーザの入力を秘匿したまま実行可能なニューラルネットワークに関して，計算量を抑えつつ，精度の劣化も抑えた秘匿 BinarizedCNN を提案した．提案方式では，BNN における BatchNormalization と Activation（Sign）をまとめて実行することで，全ての演算を等価の整数演算で置き換えた．これにより，つまり小数演算を整数演算で近似することに伴う精度劣化をなくし，高精度を維持することが出来た．また，秘密計算と相性が良いニューラルネットワークアーキテクチャを探索し，少ない計算量で精度を向上させることが出来た．

参考文献

[1] Assi Barak, Daniel Escudero, Anders Dalskov, and Marcel Keller. Secure evaluation of quantized neural networks, 2019. IACR ePrint Archive, https://eprint.iacr.org/2019/131.

[2] Octavian Catrina and Sebastiaan de Hoogh. Improved primitives for secure multiparty integer computation. In *Proc. of SCN 2010*, volume 6280 of *LNCS*, pages 182–199. Springer, 2010.

[3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016. arXiv preprint, https://arxiv.org/abs/1602.02830.

[4] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, Baltimore, MD, August 2018. USENIX Association.

[5] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *Proc. of IEEE S&P*, pages 19–38. IEEE, 2017.

[6] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. Xonn: Xnor-based oblivious deep neural network inference. In *USENIX Security*, August 2019.

[7] Adi Shamir. How to share a secret. *Communication of the ACM*, 22(11):612–613, 1979.

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

[9] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: Efficient and private neural network training, 2018. https://eprint.iacr.org/2018/442.

[10] 西田直央, 大庭達海, 加藤遼, 海上勇二, 山田翔太, アッタラパドゥンナッタポン, 照屋唯紀, 松田隆宏, and 花岡悟一郎. Binarized neural networks を用いた秘匿予測プロトビル. In コンピュータセキュリティシンポジウム, 2017.