準同型暗号上での近似活性化関数を用いた畳み込みニューラルネットワーク推論の検討
－精度改善に向けて－

石山 琢己† 森澤 竣‡ 鈴木 拓也‡ 石巻 優‡ 山名 早人§,¶

†早稲田大学基幹理工学部 〒169-8555 東京都新宿区大久保 3-4-1
‡早稲田大学大学院基幹理工学研究科 〒169-8555 東京都新宿区大久保 3-4-1
§早稲田大学理工学術院 〒169-8555 東京都新宿区大久保 3-4-1
¶国立情報学研究所 〒101-8430 東京都千代田区一ツ橋 2-1-2

E-mail: †‡§{takumi, hiroshun, t-suzuki, yuishi, yamana}@yama.info.waseda.ac.jp

あらまし 近年，クライアントのデータを利用して，クラウドサーバで機械学習モデルの訓練・推論処理を行う MLaaS (Machine Learning as a Service)が注目を集めている．しかし，金融情報や医用画像など機密データを扱う場合，プライバシーや情報漏洩の危険性の観点から MLaaS を提供しているクラウドサーバにデータを平文でアップロードすることは難しい．MLaaS で機密データを扱う方法の一つとして，暗号化されたデータを復号せずに演算可能な準同型暗号を用いるアプローチが挙げられる．ビット単位で暗号化する準同型暗号方式は，任意の演算を行えるが実行時間が長いという欠点がある．そこで，より実行時間が短い，整数や複素数を暗号化する準同型暗号方式を用いる手法がある．ただし，比較や除算，指数を伴う関数を扱うことができないため，深層学習において一般的な活性化関数である ReLU や Sigmoid を利用することができない．したがって，準同型暗号を介した機械学習の推論処理に関する既存の研究では，活性化関数として二乗関数を使用する場合が多く，分類精度が低くなってしまう問題がある．そこで本研究では，準同型暗号上での畳み込みニューラルネットワークの推論処理において，ReLU の代わりとして提案されている Swish を多項式近似した活性化関数を使用し，さらに深層学習において一般的に使われている手法である Batch Normalization を適用することで分類精度の向上を目指した．評価実験では，準同型暗号ライブラリ SEAL を用いて畳み込みニューラルネットワークの推論処理を実装し，MNIST データセットと CIFAR-10 データセットを用いて分類精度と実行時間の評価を行った．MNIST データセットでは 99.21%，CIFAR-10 データセットでは 80.47%の分類精度を達成した．

キーワード 準同型暗号，プライバシー保護機械学習

1. はじめに

近年，クライアントのデータを利用して，クラウドサーバで機械学習モデルの訓練・推論処理を行う MLaaS (Machine Learning as a Service)が注目を集めている．しかし，金融情報や医用画像などの機密データを扱う場合，プライバシーや情報漏洩の危険性の観点から MLaaS を提供しているクラウドサーバにデータを平文でアップロードすることは難しい．そこで，プライバシーを保護しながら機械学習モデルの訓練・推論処理を行う「プライバシー保護機械学習」(PPML: Privacy Preserving Machine Learning)が盛んに研究されている．

PPML の研究において，主に使用されている技術としては秘匿回路[1]や秘密分散[2]を利用し複数のパーティ間で秘匿しながら通信を行い関数評価する Multi Party Computation(MPC)，データを暗号化状態のまま演算を行うことができる準同型暗号(HE: Homomorphic Encryption)[3, 4]，Intel SGX[5] といったエンクレーブと呼ばれる暗号的に保護されたハードウェア領域でプライベートなデータを保護しつつコードを実行できる Trusted Computing Base(TCB)[6]が挙げられる．MPC を用いたアプローチは計算コストが小さいが，パーティ間で多くのインタラクションが必要なため通信コストが大きい．準同型暗号を用いたアプローチはクライアント・サーバーモデルでのプライバシー保護推論を想定した際，クライアントとサーバ間のインタラクションが 1 往復のみと少ないため，MPC アプローチと比較し大幅に通信コストが小さい．その反面，暗号化したまま計算を行うため計算コストが大きくなってしまう．TCB を用いたアプローチは計算コストや通信コストについて効率的だが，Spectre attacks[7]といった攻撃が発見されている．

各技術について上記のような利点・欠点があるが，PPML の実用化に向けて考えられる有効的な解決法の 1 つとして，準同型暗号を用いたアプローチのボトルネックとなっている計算コストを GPU の使用により削減することが考えられる．実際，深層学習において GPU を活用することで，CPU と比べて 6 倍以上高速化が達成できている[8]ため，準同型暗号を利用した PPML と組み合わせることは有望である．ビット単位

で暗号化する準同型暗号方式は，任意の演算を行えるが実行時間が大きいという欠点がある．そこで，より実行時間が短い，整数や複素数を暗号化する準同型暗号方式を用いる手法がある．ただし，深層学習における活性化関数でよく使われる ReLU や Sigmoid のような比較や除算，指数演算を伴う関数を扱うことができないといった制約が存在する．そこで準同型暗号を用いた PPML の近年の研究では，多項式で上記のような関数を近似するアプローチ[9, 10, 11, 12]やビット単位で暗号化し，バイナリゲートで構成された任意のブール回路を評価できる完全準同型暗号(FHE: Fully Homomorphic Encryption)方式である TFHE とバイナリニューラルネットワークを組み合わせることで高速化を図っているアプローチ[13, 14]などがある．

2018 年に Badawi らは，準同型暗号を使った畳み込みニューラルネットワーク(CNN: Convolutional Neural Network)に対し，初めて GPU を使って高速化を行った[15]．MNIST データセット[16]では十分な分類精度(99%)が出ているが，CIFAR-10 データセット[17]では分類精度が低い結果(77.55%)となっている．これは活性化関数に 2 乗関数を使っていることが主な原因であると考えられる．

本稿では，Badawi らの結果[15]を受けて準同型暗号上での CNN における精度改善に向けた新たな手法を提案する．提案手法は，CNN における活性化関数において ReLU の代わりとして提案されている Swish[18]を多項式近似した活性化関数を使用し，さらに深層学習において一般的に使われている手法である Batch Normalization[19]を適用することで分類精度の向上を目指す．

本稿では，以下の構成を取る．第 2 節で準同型暗号について述べ，第 3 節では準同型暗号上での畳み込みニューラルネットワーク推論の関連研究について述べる．第 4 節で提案手法について説明し，第 5 節で評価実験の結果，及び結果の考察について述べる．最後に，第 6 節でまとめを行う．

2. 準同型暗号の概要

本節では準同型暗号の概要を説明する．準同型暗号とは，暗号文を復号することなく暗号化したまま加算(準同型加算)や乗算(準同型乗算)を行うことができる暗号方式である．準同型暗号では，平文にランダムなノイズを加えて暗号化することで元のデータの解読困難性を担保している．このノイズは準同型演算を行う度に増加[1]し，一定の値を超えると正しく復号することができない．

準同型暗号は以下のように 4 つに分類できる．

- Partially Homomorphic Encryption (PHE)
  - 準同型加算か準同型乗算のどちらかの演算のみが可能である準同型暗号
- Somewhat Homomorphic Encryption (SHE)
  - 準同型加算と数回程度の準同型乗算乗算が可能である準同型暗号
- Leveled Homomorphic Encryption (LHE)
  - 準同型加算と事前に決められた回数までの準同型乗算が可能である準同型暗号
  - 暗号文に対し適用できる準同型乗算の上限回数をレベルと呼ぶ
- Fully Homomorphic Encryption (FHE)
  - 任意の回数の準同型加算と準同型乗算を可能とする準同型暗号
  - Bootstrapping と呼ばれる準同型に復号回路を評価する処理により暗号文のノイズを削減することができ，任意の回数の乗算が可能となっている
  - つまり，SHE または LHE に Bootstrapping を適用することで FHE となる

現時点では，FHE における Bootstrapping は時間計算量，空間計算量の両方が大きく，実用的なアプリケーションには適用しがたい．したがって，CNN のような機械学習のアプリケーションに準同型暗号を適用する場合は，あらかじめ乗算の回数を決定することができる LHE の使用が適している．また，LHE におけるレベルは，大きいほど実行速度が低下しメモリ使用量が増加してしまうため，可能な限りレベルを小さくする(乗算の回数を減らす)ことが求められる．

LHE である代表的な準同型暗号方式は BGV 方式[22]や BFV 方式[23, 24, 25]，CKKS 方式[4, 26]などが挙げられる．BGV 方式と BFV 方式は整数を表すことに適している．本研究では近似的に実数を表すことができ，機械学習アプリケーションに適している CKKS 方式を用いる．

これらの準同型暗号方式では，パッキング [27]と呼ばれる 1 つの暗号文に複数のデータを格納できる手法を扱うことができる．パッキングを用いることで SIMD 演算が可能となりスループットの向上につながる．本研究においてもパッキングを適用する．

3. 関連研究

本節では，近年の準同型暗号を用いた CNN の推論処理において，活性化関数を多項式近似している研究について説明する．なお，関連研究および本研究は準

[1] 準同型乗算によるノイズ増加量が特に大きい．

同型暗号上での推論処理においてクラウドサーバ側が訓練済みモデルを既に持っていることを仮定している．図 3.1 に概要を示す．主な手順は以下の①～⑥である．クラウドサーバは暗号化されたデータを復号することがないため，クライアントが所有するデータに関するプライバシーは保護される．

① クライアントが推論対象のデータを公開鍵で暗号化する
② クライアントが暗号化されたデータをクラウドサーバへ送る
③ クラウドサーバが訓練済みモデルを使って暗号化されたまま推論処理する
④ クラウドサーバが暗号化された推論結果を得る
⑤ クラウドサーバが暗号化された推論結果をクライアントへ送る
⑥ クライアントが暗号化された推論結果を秘密鍵で復号し，推論結果を得る

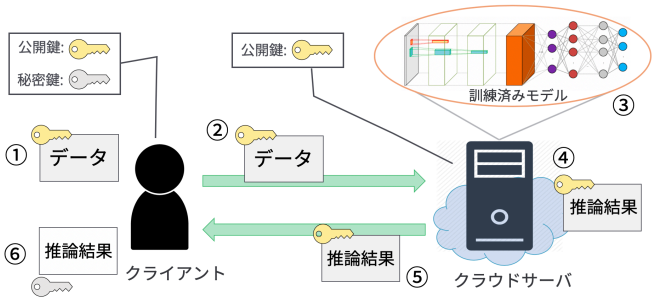

図 3.1 準同型暗号を使った推論処理

Dowlin ら[9]は，準同型暗号上で CNN の推論処理を最初に行った．活性化関数は二乗関数を使用しており，プーリングでは scaled mean pooling 使用している．scaled mean pooling はプーリングを適用するウィンドウの要素数をnとしたとき，n個の要素の総和であり，$\sum_{i=1}^{n} x_i$と表される．MNIST データセットを評価に用いており，5 層のネットワークで推論処理を行い 99%の分類精度を達成した．また，使用した準同型暗号方式は LHE の 1 つである YASHE 方式[20]である．ただし，この YASHE 方式は Albrecht ら[21]によって提案された攻撃手法より安全ではないため現在は使用するべきではない．

Chabanne ら[10]は，最小二乗法によるカーブフィッティングを用いて，ReLU を様々な次数まで多項式近似した．また，活性化関数の前に Batch Normalization を適用している．プーリングでは average pooling を使用している．average pooling はプーリングを適用するウィンドウの要素数をnとしたとき，n個の要素の平均であり，$\frac{1}{n}\sum_{i=1}^{n} x_i$と表される．

MNIST データセットを評価に用いており，式(3.1)に示した ReLU の 4 次までの多項式近似を活性化関数として使用した結果，24 層のネットワークで 97.91%の分類精度を達成した．また，使用した準同型暗号方式は BGV 方式である．

$$f(x) = 0.1500 + 0.5012x + 0.2981x^2 - 0.0004x^3 - 0.0388x^4 \quad (3.1)$$

Jiang ら[11]は，活性化関数に二乗関数を使用しており，プーリングは使用していない．MNIST データセットを評価に用いており，5 層のネットワークで 98.1%の分類精度を達成した．また，使用した準同型暗号方式は CKKS 方式である．

Hesamifard ら[12]は，ReLU をそのまま近似するのではなく，ReLU の導関数を近似した．具体的には，ReLU の導関数であるステップ関数と形状が近いグモイド関数を多項式で近似し，その多項式の積分を活性化関数として使用した．プーリングでは scaled mean pooling を使用している．MNIST データセットを評価に用いており，6 層のネットワークで 99.25%の分類精度を達成した．また，使用した準同型暗号方式は BGV 方式である．

Badawi ら[15]は，準同型暗号を使った CNN の推論に対し，初めて GPU を使って高速化を図った．活性化関数に二乗関数を使用しており，プーリングでは average pooling を使用している．MNIST データセットと CIFAR-10 データセットに対してそれぞれ 99%，77.55%の分類精度を達成した．また，使用した準同型暗号方式は BFV 方式である．パッキングを用いて推論対象の複数の画像の同一ピクセルを単一の暗号文に格納し，SIMD 形式で同時に推論処理を行っている．

既存研究の問題点の 1 つとして，MNIST データセットに対しては概ね分類精度が高いが，CIFAR-10 データセットに対しての分類精度が低いという点が挙げられる．例えば，本節で挙げた関連研究で唯一 CIFAR-10 データセットに対して評価を行っている[15]では，分類精度は 77.55%となっている．既存の深層学習の研究において，CIFAR-10 データセットの分類精度は概ね 90%を超えている[2]ため，77.55%は低い．分類精度が低い原因として，活性化関数に二乗関数を使用していることが考えられる．したがって，CIFAR-10 データセットのようなデータセットに対しても十分な分類精度を得ることができる手法が求められる．

[2] https://benchmarks.ai/cifar-10

4. 分類精度向上に向けた準同型暗号上での推論処理の提案

本節では，Badawi らの CIFAR-10 データセットに対する結果[15]を受けて，準同型暗号上での CNN の推論処理における分類精度向上に向けての提案手法について述べる．4.1 項では，ReLU の代わりとして提案されている Swish の多項式近似についての説明を行う．4.2 項では，深層学習において一般的に使われている手法である Batch Normalization の適用について説明を行う．4.3 項では，消費レベル削減の工夫について述べる．

4.1 活性化関数 Swish の多項式近似

Swish[18]は，ReLU($f(x) = \max(0, x)$)に代わる活性化関数の研究において発見された汎用性の高い活性化関数であり，式 4.1 で表される．式 4.1 からわかるように，Swish は Sigmoid($f(x) = \frac{1}{1+e^{-x}}$)に$x$をかけたものである．ReLU と Swish の比較を図 4.1 に示す．

$$f(x) = \frac{x}{1+e^{-x}} \tag{4.1}$$

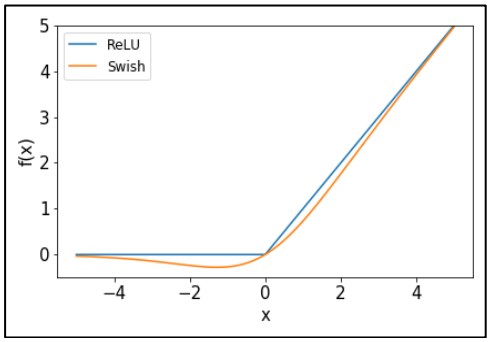

図 4.1 ReLU と Swish の比較

後述の Batch Normalization による標準化を活性化関数の直前に適用することにより，活性化関数の入力値はガウス分布に従う．よって，0 付近の定義域において，近似対象である Swish との誤差が最小化できればよいため，x軸の範囲$[-4,4]$と$[-6,6]$の Swish に対して，多項式の次数を 4 次とし多項式近似を行った．python の scipy ライブラリ[3]で提供されている leastsq 関数[4]を使用して多項式近似した結果を以下の表 4.1 に示す．これらの 4 次の多項式を準同型暗号上で評価するのに必要なレベルは 3 である．図 4.2 に Swish と多項式近似した Swish の比較を示す．

表 4.1 活性化関数 Swish の多項式近似結果

近似対象のx軸の範囲	Swish の多項式近似
−4~4	$0.03347 + 0.5 \times x + 0.19566 \times x^2 - 0.005075 \times x^4$
−6~6	$0.1198 + 0.5 \times x + 0.1473 \times x^2 - 0.002012 \times x^4$

[3] https://www.scipy.org

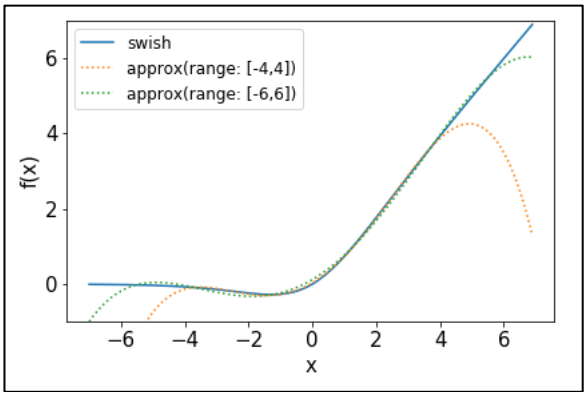

図 4.2 Swish と多項式近似した Swish の比較

4.2 Batch Normalization の適用

Batch Normalization[19]は，深層学習において一般的に用いられている手法の一つであり，ミニバッチ内の要素(チャンネルやユニット)ごとの層の出力の平均と分散を用いて標準化することで，ネットワーク内部の変数の分布(内部共変量シフト)が大きく変わるのを防ぐ．分類精度を向上させる効果や学習の収束が早くなる効果がある．

推論時の Batch Normalization の処理は訓練時に得られる重みパラメータをW_{BN}，バイアスパラメータをB_{BN}としたとき，式 4.2 で表される(入力をx，出力をyとする)．式 4.2 より，準同型暗号上での推論における Batch Normalization の処理はレベルを 1 消費することがわかる．

$$y = W_{BN} \times x + B_{BN} \tag{4.2}$$

4.3 消費レベル削減の工夫

4.1 項で示した Swish の 4 次の多項式近似はレベルを 3 消費する．また，提案手法では活性化関数の直前に Batch Normalization を適用するためさらに 1 消費する．したがって合計 4 レベルの消費が発生し，Badawi ら[15]が活性化関数として利用している二乗関数の消費レベル 1 と比較すると，提案手法は活性化関数を 1 回評価するごとに 3 つ多くレベルを消費することになる．そこで，消費レベルを削減する(乗算の回数を減らす)ために 2 つの工夫を行う．なお，これらの工夫は深層学習における最適化において一般的に用いられており，準同型暗号上での深層学習の最適化[29, 30]においても適用されている．

1 つ目は，畳み込み層と Batch Normalization の結合である．畳み込み層において，ある$s \times s \times c$のフィルタとそのフィルタに対応する入力xに対し畳み込みを行い，出力yを得る処理を考える．訓練済みモデルから得られる畳み込み層の重みとバイアスをそれぞれ$s \times s \times c$の行列W_{Conv}とスカラーB_{Conv}とし，4.2 項の式(4.2)

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.leastsq.html

で示した Batch Normalization の定数パラメータW_{BN}, B_{BN}を用いて，畳み込み層と Batch Normalization の推論処理はまとめて式(4.3)で表すことができる．なお，kチャネル目の縦i番目横j番目の入力とフィルタの重みをそれぞれ$x^{i,j,k}, W_{Conv}^{i,j,k}$とする．式(4.3)の$W_{Conv}^{i,j,k}W_{BN}$や$B_{Conv}W_{BN}+B_{BN}$は，事前に計算しておくことができるため，必要なレベルを増やすことなく Batch Normalization を適用することができる．

$$y = W_{BN}\left(\sum_{i=0}^{s}\sum_{j=0}^{s}\sum_{k=0}^{c} W_{Conv}^{i,j,k}x^{i,j,k} + B_{Conv}\right) + B_{BN}$$

$$= \left(\sum_{i=0}^{s}\sum_{j=0}^{s}\sum_{k=0}^{c} W_{Conv}^{i,j,k}W_{BN}x^{i,j,k}\right) + (B_{Conv}W_{BN} + B_{BN}) \quad (4.3)$$

2 つ目は，Swish の多項式近似における係数の操作である．提案手法である多項式近似を用いた Swish 活性化関数は$f(x) = ax^4 + bx^2 + cx + d$と表すことができる．ここで，最高次の項の係数$a$で両辺を割り，$f'(x) = \frac{f(x)}{a}, b' = \frac{b}{a}, c' = \frac{c}{a}, d' = \frac{d}{a}$とすると，$f'(x) = x^4 + b'x^2 + c'x + d'$が得られる．この$f'$という関数の消費レベルは 2 である．そして，$f'$はもとの関数$f$を$\frac{1}{a}$倍したものであるため，活性化関数を適用した直後のプーリング層や全結合層の重みパラメータに対し事前にaをかける．

上記 2 つの工夫を適用することで，Batch Normalization と多項式近似した Swish の評価に必要なレベルをそれぞれ単純に評価した場合と比べて，2 少なくすることができる．また，二乗関数の評価と比べ，必要なレベルの増加量を 1 に抑えることができる．

5. 評価実験

本節では，提案手法に対する評価実験の手法と実験結果について述べる．

5.1 データセット

評価実験では，MNIST データセット[16]と CIFAR-10 データセット[17]を用いた．

MNIST データセットは，手書きで書かれた 0~9 の数字 1 文字を画像にしたグレースケールの28×28ピクセルの画像データと，その画像に書かれた数字を表すラベルデータから構成されるデータセットである．訓練データは 60,000 件，テストデータは 10,000 件の計 70,000 件のデータから成る．

CIFAR-10 データセットは，異なる 10 クラスの RGB カラーの32×32ピクセルの画像データと、正解ラベルデータから構成されるデータセットである．訓練データは 50,000 件，テストデータは 10,000 件の計 60,000 件のデータから成る．

5.2 ネットワークの構成

本実験で用いる CNN の構成は Badawi ら[15]と同じ構成とする．MNIST データセットに対して適用するネットワークの構成を表 5.1 に，CIFAR-10 データセットに対して適用するネットワークの構成を表 5.2 にそれぞれ示す．ただし，提案手法では Batch Normalization を各畳み込み層の直後に追加する(表 5.1 と表 5.2 において太字で示し，BN と略記)．

表 5.1 MNIST データセットに用いる CNN の構成

層の種類	説明	出力サイズ
畳み込み	5×5のフィルタ5個 ストライド$(2,2)$ パディングなし	$12 \times 12 \times 5$
活性化	活性化関数を適用	$12 \times 12 \times 5$
BN	**BN を適用**	$\mathbf{12 \times 12 \times 5}$
畳み込み	5×5のフィルタ50個 ストライド$(2,2)$ パディングなし	$4 \times 4 \times 50$
活性化	活性化関数を適用	$4 \times 4 \times 50$
BN	**BN を適用**	$\mathbf{4 \times 4 \times 50}$
全結合	10 個のニューロンへの加重和を計算	$1 \times 1 \times 10$

表 5.2 CIFAR-10 データセットに用いる CNN の構成

層の種類	説明	出力サイズ
畳み込み	3×3のフィルタ32個 ストライド$(1,1)$ パディング$(1,1)$	$32 \times 32 \times 32$
活性化	活性化関数を適用	$32 \times 32 \times 32$
BN	**BN を適用**	$\mathbf{32 \times 32 \times 32}$
プーリング	2×2のプールサイズ ストライド$(2,2)$	$16 \times 16 \times 32$
畳み込み	3×3のフィルタ64個 ストライド$(2,2)$ パディング$(1,1)$	$16 \times 16 \times 64$
活性化	活性化関数を適用	$16 \times 16 \times 64$
BN	**BN を適用**	$\mathbf{16 \times 16 \times 64}$
プーリング	2×2のプールサイズ ストライド$(2,2)$	$8 \times 8 \times 64$
畳み込み	3×3のフィルタ128個 ストライド$(2,2)$ パディング$(1,1)$	$8 \times 8 \times 128$
活性化	活性化関数を適用	$8 \times 8 \times 128$
BN	**BN を適用**	$\mathbf{8 \times 8 \times 128}$
プーリング	2×2のプールサイズ ストライド$(2,2)$	$4 \times 4 \times 128$
全結合	256 個のニューロンへの加重和を計算	$1 \times 1 \times 256$
全結合	10 個のニューロンへの加重和を計算	$1 \times 1 \times 10$

5.3 平文での機械学習モデルの訓練

Keras ライブラリ[5]を用いて，5.2 項で示したネットワークの構造に従ってモデルの訓練を行った．活性化関数は ReLU，Swish，二乗関数，表 4.1 に示した 4 次の多項式で近似した Swish(x軸の近似範囲$[-4\sim4]$と$[-6\sim6]$)の 5 通りで行った．なお，ReLU，Swish，多項式近似した Swish については，畳み込み層の直後に Batch Normalization を適用した．

5.4 平文での実験結果

研究で用いる CKKS 方式は，実数を近似するため元のデータと微小な誤差が生じ，暗号文上での推論処理により得られる分類精度は平文での分類精度より低くなることが予測される．したがって，活性化関数を近似したことによる分類精度の変化と CKKS 方式の適用による分類精度の変化のそれぞれを比較するために，まず 5.3 項で得られた訓練済みモデルを用いて，平文のテストデータに対し推論処理を行い，分類精度を測定した．表 5.3，表 5.4 に MNIST データセットと CIFAR-10 データセットそれぞれの測定結果を示す．

表 5.3 MNIST データセットに対する平文での分類精度

活性化関数	分類精度 [%]
ReLU	99.05
Swish	99.25
二乗	99.18
近似 Swish (x: $[-4,4]$)	99.16
近似 Swish (x: $[-6,6]$)	99.21

表 5.4 CIFAR-10 データセットに対する平文での分類精度

活性化関数	分類精度 [%]
ReLU	83.69
Swish	82.00
二乗	77.04
近似 Swish (x: $[-4,4]$)	80.09
近似 Swish (x: $[-6,6]$)	80.47

5.5 評価方法

CKKS 方式を実装している SEAL ライブラリ[28]を用いて準同型暗号上での CNN の推論処理を実装した．MNIST データセットと CIFAR-10 データセットのテストデータ各 10,000 件を暗号化し，暗号化されたまま訓練済みモデルを用いて推論処理を行い，分類精度を測定する．Badawi ら[15]と同様に，パッキングを用いて推論対象の複数の画像の同一ピクセルを単一の暗号文に格納し，SIMD 形式で同時に推論処理を行った．活性化関数は二乗関数と表 4.1 に示した 4 次の多項式で近似した Swish でそれぞれ評価を行い，既存手法と提案手法での分類精度，実行時間，メモリ使用量を比較する．なお，4.3 項で示した工夫を施さず畳み込み層と Batch Normalization と多項式近似した Swish をそれぞれ独立に評価する手法を提案手法①とし，4.3 項で示した畳み込み層と Batch Normalization の結合，多項式近似した Swish の係数操作の工夫を両方適用する手法を提案手法②とする．

本実験を行った計算機の CPU は Intel Xeon E7-8880 v3(2.30GHz)，CPU 数 4，1CPU あたりのコア数 18，メモリは 3TB，OS は CentOS7.6.1810 である．コンパイラは GCC7.4.0 を使用した．

5.6 評価結果

二乗関数と表 4.1 に示した 4 次の多項式で近似した Swish(x軸の近似範囲$[-4,4]$と$[-6,6]$)の 3 つの活性化関数を用いて準同型暗号上で CNN の推論処理をそれぞれ行い，得られた分類精度と推論処理時間，メモリ使用量の測定結果を示す．

5.6.1 MNIST データセットでの測定結果

MNIST データセットに対する既存手法での測定結果を表 5.5 に，提案手法①での測定結果を表 5.6 に，提案手法②での測定結果を表 5.7 に示す．また，各手法のスレッド数ごとの推論処理時間をまとめたグラフを図 5.1 に示す．なお，OpenMP によるスレッド並列を適用し，使用スレッド数 1，8，12，16，32 の 5 通りで測定を行った．

表 5.3 に示した平文での分類精度と表 5.5，5.6，5.7 に示した準同型暗号上での分類精度を比較すると，それぞれの手法において分類精度が一致しており，CKKS 方式での暗号化によって生じる元のデータとの微小な誤差は分類精度に影響がなかった．また，推論に必要なレベルと推論処理時間は比例の関係にあることや図 5.3 からマルチスレッドの適用により推論処理時間を削減できていることがわかる．例えば，区間$[-6,6]$で 4 次まで近似した Swish と Batch Normalization を用いた提案手法②において 12 スレッド使用した場合，テストデータ 10,000 件の推論実行時間は 74.2 秒であり，シングルスレッド時の実行時間である 584.6 秒と比較して約 8 倍高速化した．区間$[-6,6]$で 4 次まで多項式近似した Swish と Batch Normalization を用いた提案手法では分類精度が 99.21%であり，二乗関数を用いた既存手法での分類精度 99.18%と比べて 0.03%の精度向上を達成した．

表 5.5，5.6，5.7 より準同型暗号上での推論処理におけるメモリ使用量は必要レベルに比例していることがわかる．このメモリ使用量の大半を占めているのが，CKKS 方式でエンコードした訓練済みモデルのパラメータである．

[5] https://github.com/keras-team/keras

表 5.5 既存手法での MNIST データセットに対する測定結果

活性化関数	必要レベル	メモリ使用量 [GB]	スレッド数	分類精度 [%]	推論処理時間 [sec]
二乗関数	5	35.4	1	99.18	392.0
			8	99.18	56.3
			12	99.18	49.8
			16	99.18	32.3
			32	99.18	27.0

表 5.6 提案手法①での MNIST データセットに対する測定結果

活性化関数	必要レベル	メモリ使用量 [GB]	スレッド数	分類精度 [%]	推論処理時間 [sec]
近似 Swish (x: $[-4,4]$)	11	58.6	1	99.16	953.4
			8	99.16	138.8
			12	99.16	120.5
			16	99.16	78.9
			32	99.16	63.7
近似 Swish (x: $[-6,6]$)	11	58.6	1	99.21	950.2
			8	99.21	150.2
			12	99.21	117.4
			16	99.21	78.3
			32	99.21	66.4

表 5.7 提案手法②での MNIST データセットに対する測定結果

活性化関数	必要レベル	メモリ使用量 [GB]	スレッド数	分類精度 [%]	推論処理時間 [sec]
近似 Swish (x: $[-4,4]$)	7	44.6	1	99.16	587.0
			8	99.16	90.6
			12	99.16	71.7
			16	99.16	51.1
			32	99.16	40.4
近似 Swish (x: $[-6,6]$)	7	44.5	1	99.21	584.6
			8	99.21	86.6
			12	99.21	74.2
			16	99.21	50.3
			32	99.21	42.5

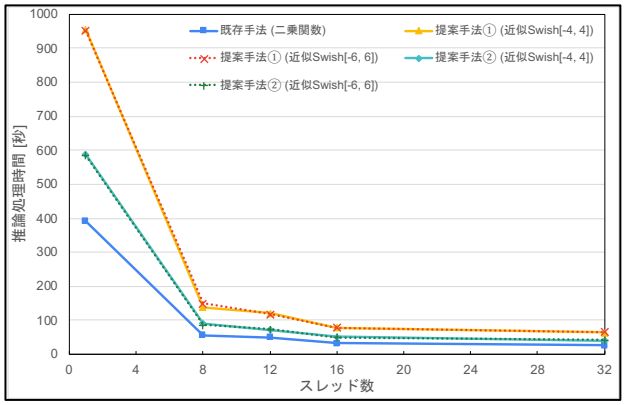

図 5.1 MNIST データセットに対する推論処理時間の測定結果

5.6.2 CIFAR-10 データセットでの測定結果

CIFAR-10 データセットに対する測定結果を表 5.8 に示す．なお，CIFAR-10 データセットでは実行時間とメモリ使用量の都合上，既存手法と提案手法②に対し 72 スレッドでのみ測定を行った．

表 5.4 に示した平文での分類精度と表 5.8 に示した準同型暗号上での分類精度を比較すると，区間$[-6,6]$で近似した Swish を利用した提案手法②では分類精度が 0.01%低下しており，既存手法と区間$[-4,4]$で近似した Swish を利用した提案手法②では分類精度が一致している．よって，CKKS 方式での暗号化によって生じる元のデータとの微小な誤差は分類精度にほとんど影響がないと考えられる．区間$[-6,6]$で 4 次まで多項式近似した Swish を用いた提案手法②では分類精度が 80.47%であり，二乗関数を用いた既存手法での分類精度 76.37%と比べて 4.1%の精度向上を達成した．

また，テストデータ 10,000 件の暗号文上での推論処理時間は 72 スレッドを使用して，既存手法が 3,577 秒，提案手法がそれぞれ 5,059 秒と 5,171 秒であり，提案手法が既存手法と比べ約 1.4 倍の時間を要した．

MNIST データセットでの測定結果と同様，必要レベルの大きい提案手法の方が既存手法と比べてメモリ使用量が大きい．また，CIFAR-10 データセットで使用した訓練済みモデルのパラメータ数は MNIST データセットで使用した訓練済みモデルのパラメータ数の約 43 倍であり，既存手法で 2.31TB，提案手法で 2.82TB のメモリ使用量であった．

表 5.8 CIFAR-10 データセットに対する測定結果 (72 スレッド使用)

手法	活性化関数	必要レベル	メモリ使用量 [TB]	分類精度 [%]	推論処理時間 [sec]
既存手法	二乗関数	8	2.31	76.37	3,576.7
提案手法②	近似 Swish (x: $[-4,4]$)	11	2.82	80.08	5,059.1
	近似 Swish (x: $[-6,6]$)	11	2.82	80.47	5,171.1

6. おわりに

本稿では，準同型暗号上での CNN 推論処理において，分類精度を向上させる手法を提案した．具体的には，従来の研究において活性化関数として用いられることの多かった二乗関数ではなく，ReLU の代替として発見された Swish を 4 次の多項式で近似した関数を使い，さらに Batch Normalization を組み合わせた．また，4 次の多項式近似の評価と Batch Normalization の適用による消費レベル(乗算回数)の増加を抑えるた

めに，CNN における層の結合や Swish の近似多項式の係数操作といった工夫を行った．

評価の結果，MNIST データセットに対する分類精度として 99.21%を達成し，二乗関数を活性化関数として利用する既存研究の手法と比較して，0.03%の精度向上を確認した．また，CIFAR-10 データセットに対しては分類精度 80.47%を達成し，二乗関数を活性化関数として利用する既存研究の手法と比較して，4.1%の精度向上を確認した．さらに，MNIST データセットでは様々なスレッド数で推論を実行し，32 スレッドで 13.8 倍の高速化を達成した．CIFAR-10 データセットでは 72 スレッドで 10,000 件の画像分類を行うのに要した時間は 5,171 秒であり，二乗関数を利用する従来手法の 1.4 倍の実行時間となった．

今後の課題としては，推論処理におけるメモリ使用量の削減，GPU により推論処理を高速化すること，ReLU や max pooling を評価できるような新たなプロトコルを考案すること，より深いネットワークモデルへの適用などが挙げられる．

謝　　辞

本研究の一部は，JST（CREST，JPMJCR1503）及び NICT（Japan–US Network Opportunity 2 (JUNO2)）の支援を受けたものである．

参　考　文　献

[1] A. C.-C. Yao, “How to Generate and Exchange Secrets,” in *Proc. of the 27th IEEE Symposium on FOCS*, pp. 162–167, 1986.
[2] A. Shamir, “How to Share a Secret,” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[3] C. Gentry, “Fully Homomorphic Encryption Using Ideal Lattices,” in *Proc. of the 41st Ann. ACM STOC*, pp. 169–178, 2009.
[4] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic Encryption for Arithmetic of Approximate Numbers,” in *Proc. of ASIACRYPT 2017*, pp. 409–437, 2017.
[5] F. McKeen, I. Alexandrovich, A. Berenzon, C. Rozas, H. Shafi, V. Shanbhogue and U. Savagaonkar, “Innovative Instructions and Software Model for Isolated Execution,” in *Proc. of the 2nd International Workshop on HASP*, no. 10, 2013.
[6] J. Rushby, “Design and Verification of Secure Systems,” in *Proc. ACM OSR*, vol. 15, no. 5, pp. 12–21, 1981.
[7] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz and Y. Yarom, “Spectre attacks: Exploiting speculative execution,” in *Proc. of IEEE S&P*, vol. 2019-May, pp. 1–19, 2019.
[8] NVIDIA blog, "NVIDIA Propels Deep Learning with TITAN X, New DIGITS Training System and DevBox," https://blogs.nvidia.com/blog/2015/03/17/digits-devbox/, accessed Jan. 9. 2020.
[9] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “CryptoNets: applying neural networks to encrypted data with high throughput and accuracy,” in *Proc. of the 33rd ICML*, vol. 48. JMLR.org, pp. 201–210, 2016.
[10] H. Chabanne, A. De Wargny, J. Milgram, C. Morel, and E. Prouff, “Privacy-Preserving Classification on Deep Neural Network,” *IACR Cryptology ePrint Archive*, Report 2017/035, 2017.
[11] X. Jiang, K. Lauter, M. Kim, and Y. Song, “Secure outsourced matrix computation and application to neural networks,” in *Proc. of ACM CCS*, pp. 1209–1222, 2018.
[12] E. Hesamifard, H. Takabi, and M. Ghasemi, “Deep neural networks classification over encrypted data,” in *Proc. of the 9th ACM DASP*, pp. 97–108, 2019.
[13] F. Bourse, M. Minelli, M. Minihold and P. Paillier, “Fast Homomorphic Evaluation of Deep Discretized Neural Networks,” *IACR Cryptology ePrint Archive*, Report 2017/1114, 2017.
[14] A. Sanyal, M. J. Kusner, A. Gascón, and V. Kanade, “TAPAS: Tricks to Accelerate (encrypted) Prediction As a Service,” *arXiv preprint arXiv:1806.03461*, 2018.
[15] A. Badawi, J. Chao, J. Lin, C. Mun, J. Sim, B. Tan, X. Nan, K. Aung and V. Chandrasekhar, “The AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data with GPUs,” *IACR Cryptology ePrint Archive*, Report 2018/1056, 2018.
[16] Y. LeCun and C. Cortes, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, accessed Jan. 9. 2020.
[17] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian Institute for Advanced Research)," www.cs.toronto.edu/~kriz/cifar.html, accessed Jan. 9. 2020.
[18] P. Ramachandran, B. Zoph, and Q. V Le Google Brain, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.
[19] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. of the 32nd ICML*, vol. 1, pp. 448–456, 2015.
[20] K. and L. J. and N. M. Bos Joppe W. and Lauter, “Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme,” in *Cryptography and Coding*, LNCS, vol. 8308, pp. 45–64, 2013.
[21] M. Albrecht, S. Bai, and L. Ducas, “A Subfield Lattice Attack on Overstretched NTRU Assumptions,” in *Proc. of CRYPTO 2016*, pp. 153–178, 2016.
[22] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(Leveled) Fully Homomorphic Encryption without Bootstrapping,” in *Proc. of the 3rd ITCS*, pp. 309–325, 2012.
[23] Z. Brakerski, “Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP,” in *Proc. of CRYPTO 2012*, vol. 7417, pp. 868–886, 2012.
[24] J. Fan and F. Vercauteren, “Somewhat Practical Fully Homomorphic Encryption,” *IACR Cryptology ePrint Archive*, Report 2012/144, 2012.
[25] J. and H. M. A. and Z. V. Bajard Jean-Claude and Eynard, “A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes,” in *Proc. of SAC 2016*, pp. 423–442, 2016.
[26] Cheon Jung Hee, K. and Han, and K. Andrey, and K. Miran, and and S. Yongsoo, “A Full RNS Variant of Approximate Homomorphic Encryption,” in *Proc. of SAC 2018*, pp. 347–368, 2018.
[27] N. P. Smart and F. Vercauteren, “Fully Homomorphic SIMD Operations,” *Designs, Codes and Cryptography*, vol. 71, no. 1, pp. 57–81, 2014.
[28] Microsoft, “Microsoft SEAL (release 3.4),” https://github.com/Microsoft/SEAL, accessed Jan. 9. 2020.
[29] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, “nGraph-HE: A Graph Compiler for Deep Learning on Homomorphically Encrypted Data,” *arXiv preprint arXiv:1810.10121*, 2019.
[30] F. Boemer, R. Cammarota, A. Costache, and C. Wierzynski, “nGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data,” *IACR Cryptology ePrint Archive*, Report 2019/947, 2019.