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に対する Channel Pruning の適用

石山 琢己† 鈴木 拓也† 山名 早人‡

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

あらまし 近年, クラウド上で機械学習モデルの訓練や推論処理を提供する MLaaS (Machine Learning as a Service)が普及しつつある．しかし，金融情報や医用画像といった機密データを扱う場合には，情報漏えいのリスクが生じる．このような外部で機密データを扱う上でのプライバシー保護とデータ利活用を両立する技術の一つとして，暗号化されたデータを復号することなく演算可能な準同型暗号が挙げられる．我々の以前の研究では，準同型暗号上で畳み込みニューラルネットワークの推論を評価したものの，レイテンシが長いという問題があった．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において、レイテンシの短縮を目指す．具体的には，推論対象データのチャネルごとに暗号化を適用し，さらに機械学習モデルに対し Channel Pruning を事前に適用することで準同型演算の回数を減らす．評価実験では，MNIST データセットと CIFAR-10 データセットを用いて分類精度及び推論レイテンシを評価した．結果，MNIST では 0.60～0.97 秒（精度 96.58～99.52%），CIFAR-10 では 8.1～12.9 秒（精度 66.52～80.96%）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，セキュリティ，畳み込みニューラルネットワーク，推論処理

1. はじめに

近年，クラウド上で機械学習モデルの訓練や推論処理を提供する MLaaS(Machine Learning as a Service)が普及しつつある．しかし，金融情報や医用画像などの機密情報を扱う場合，MLaaS を提供するクラウドサーバにデータを平文で保存することは，情報漏えいやプライバシーの観点からリスクが伴う．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や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，代表的に使用されている技術の 1 つが，データを暗号化したまま演算可能な暗号方式の準同型暗号[1], [2]である．本研究では，図 1.1 に示すような，準同型暗号のみを利用した「サーバでの処理中に通信を必要としない推論プロトコル」を対象とする．また，サーバ側が事前に学習済みモデルを保持していることを想定する．

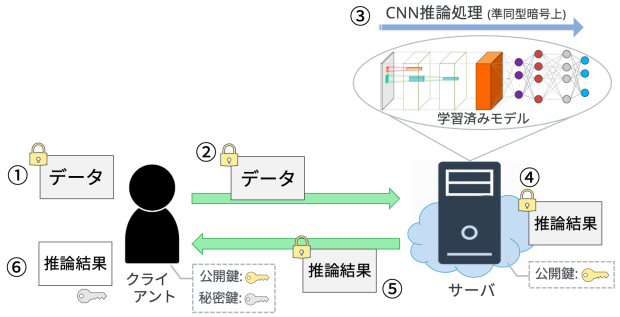

図 1.1 準同型暗号を利用したセキュアな CNN 推論プロトコル

準同型暗号は，暗号化されたデータに対して加算や乗算を評価することができる．しかし，暗号化されていない通常のデータ上での計算と比較して，準同型暗号の時間計算量・空間計算量が大きい．また，整数を暗号化可能な方式や，本研究で使用する複素数を暗号化可能な Cheon-Kim-Kim-Song (CKKS)方式[2], [3]では，比較や除算を処理することができない．そのため，深層学習において頻繁に用いられる Rectified Linear Unit (ReLU)やシグモイド関数, max pooling を扱うことができない．従来研究では，活性化関数の多項式近似への代替や，max pooling から average pooling への代替が行われている．特に，多項式近似した活性化関数の使用は分類精度の低下につながる．上記の欠点を踏まえた上で，実行パフォーマンスの向上や分類精度の改善を達成することが準同型暗号を利用した PPML の研究における目標の一つである．

本研究では，準同型暗号上での CNN(Convolutional Neural Network)推論において，多少の分類精度の低下を許容しつつ，レイテンシの短縮を目指す．レイテンシの短縮に向けて，パッキング[4]と呼ばれる，複数の平文を単一の平文または暗号文に格納する手法を活用する．具体的には，筆者らの以前の研究[5]では，スループット指標を重視し，推論対象の複数画像における同位置なピクセルを単一の暗号文に格納する方法(以下, Batch-axis Packing と呼ぶ)を採用した．これに対し，本稿では，同一画像のチャネルごとに暗号化する方法(以下，Channel-wise Packing と呼ぶ)を採用し，さらに

学習済み CNN モデルに対し事前に Channel Pruning[6], [7]と呼ばれるチャネル・フィルタ単位でのモデル圧縮手法を適用することで，推論処理中に必要な暗号文の数や準同型演算(暗号文上での加算や乗算)の数を減らす．

本稿は以下の構成である．第 2 節で準同型暗号の概要を説明し，第 3 節で関連研究について述べる．第 4 節において提案手法を説明し，第 5 節で評価実験を行う．最後に，第 6 節でまとめる．

2. 準同型暗号の概要

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．

本稿では，準同型暗号の一方式である CKKS 方式[2], [3]を用いる．CKKS 方式は，固定小数点数で表現された複素数を暗号化することができ（実数を扱うことができ），機械学習アプリケーションに適している．また，任意回数の準同型加算と事前に決められた回数(暗号文の初期レベルL)の準同型乗算が実行可能な Leveled 準同型暗号に分類できる．Leveled 準同型暗号において，ある暗号文に対し適用できる準同型乗算の上限回数をその暗号文のレベルと呼ぶ．このレベルは，大きいほど実行時間やデータサイズが増加するため，可能な限り小さくする(アプリケーションにおける準同型乗算の回数を減らす)ことが求められる．

CKKS 方式では，パッキング[4]をサポートしている．パッキングとは複数の平文をエンコードし 1 つにまとめ，そのエンコードした平文ベクトルを単一の暗号文に暗号化する技術である．具体的には，CKKS 方式の場合では，準同型暗号のパラメータの 1 つである多項式環の次数Nに対して，$N/2$個の平文をエンコードし，暗号化することで単一の暗号文に暗号化する技術である．パッキングにより，パッキングされたベクトル(平文，あるいは暗号文)の各要素(スロット)に対して，SIMD (Single Instruction, Multiple Data)形式の演算を実行することができる．つまり，2 つの暗号文同士，または暗号文とエンコードした平文ベクトルに対し，単一の演算でスロットごとに加算や乗算を同時に適用できる．CKKS 方式では，準同型加算や準同型乗算の他に，暗号文のスロットを左方向または右方向に指定したステップ数だけ循環的に回転させる *rotation* 操作が可能である．以降，ある暗号文vに対して右にnステップ$rotation$を適用する操作を$v >> n$，左にnステップ$rotation$を適用する操作を$v << n$と表す．また，暗号文のパラメータを調整するために$relinearization$や$rescaling$と呼ばれる準同型演算を適宜実行する必要がある．なお，$rotation$や$relinearization$，$rescaling$は，準同型加算や準同型乗算と比べて実行時間が長い．

3. 関連研究

Dowlin ら[8]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 を使用している．scaled mean pooling はプーリングを適用するカーネルの要素数を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 データセットのみを評価に用いており，2 層の CNN で推論処理を行い，98.95%の分類精度であった．また，Batch-axis Packing を使用しており，1 度に 4,096 枚の画像を推論できるが，レイテンシは 250 秒であった．

Chabanne ら[9]は，Dowlin らの結果に対し，活性化関数として二乗関数の使用は，層が深い場合に分類精度が低くなる問題を指摘した．そこで，最小二乗法によるカーブフィッティングを用いて，ReLU を様々な次数で多項式近似し，活性化関数の前に Batch Normalization を適用することで，分類精度の低下を抑制した．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カーネルの要素数を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 データセットのみを評価に用いており，4 次の多項式近似 ReLU を活性化関数として使用した結果，24 層の CNN で 99.30%の分類精度を達成した．しかし，レイテンシやスループットに関しては報告されていない．

Badawi ら[10]は，準同型暗号上での CNN 推論において，初めて GPU を使用して高速化を行った．活性化関数は二乗関数を使用しており，プーリングは average pooling を用いている．MNIST データセットに対し，3 層の CNN を用いて評価を行い，5.16 秒のレイテンシを達成した(分類精度は 99%)．また，CIFAR-10 データセットに対しては，5 層の CNN を用いて評価を行い，304.43 秒のレイテンシを達成した(分類精度は 77.55%)．Batch-axis Packing を使用しており，1 度に 8,192 枚の画像を推論することができる．

筆者ら[5]は，Badawi らの研究を受け，CIFAR-10 データセットに対する分類精度を向上させるために，ReLU や Swish を 2 次及び 4 次の多項式で近似し，Chabanne ら同様に，活性化関数の前に Batch Normalization を適用した．また，多項式近似や Batch Normalization の活用による消費レベルの増加を抑えるために，既存の深層学習における最適化や準同型暗号上での深層学習グラフコンパイラ[11]で使われている「畳み込み層と Batch Normalization の結合」や「多項式近似活性化関数及び average pooling の係数操作」といった工夫も適用した．Badawi らと同じネットワーク構成で評価を行い，MNIST データセットでは 2 次の多

項式近似 Swish を用いて 99.29%，CIFAR-10 データセットでは 4 次の多項式近似 ReLU を用いて 81.06%の分類精度を達成した．しかし，1 回の推論処理において，MNIST データセットに用いた CNN では実行時間が 21 秒(16 スレッド使用)，メモリ使用量が 16.2GB であり，CIFAR-10 データセットに用いた CNN では実行時間が 26 分(72 スレッド使用)，メモリ使用量が 1.41TB とレイテンシが長く，メモリ使用量も大きい結果であった．なお，Batch-axis Packing を使用しており，1 度に 8,192 枚の画像を推論することができる．

Chou ら[12]は，準同型暗号上での CNN 推論を高速化するために，Pruning や Quantization といった深層学習の圧縮手法を活用した．また，活性化関数として，ReLU と Swish，Softplus の 3 種類を 2 次の多項式で近似したものを使用した．MNIST データセットに対し，Pruning と Quantization を適用した 4 層の CNN を用いて評価を行い，45.7 秒のレイテンシを達成した．なお，活性化関数として 2 次の多項式で近似した Swish を用いており，分類精度は 98.71%である (Pruning 及び Quantization 適用前の分類精度は 99.10%)．また，CIFAR-10 データセットに対し，Pruning と Quantization を適用した 8 層の CNN を用いて評価を行い，レイテンシは 22,372 秒(6 時間 12 分 52 秒)，分類精度は 76.72%である．なお，この論文ではパッキングが使用されていない．

準同型暗号上での CNN 推論において，既存研究では，GPU を用いたハードウェアによる高速化や，Pruning や Quantization といった深層学習の圧縮手法を適用することによる高速化が行われている．しかし，推論レイテンシを短くすることが困難であり(特に CIFAR-10 データセットでのレイテンシ)，Chou ら[12]は Pruning や Quantization により高速化を達成したもののパッキングを活用していない．また，Badawi ら[10]や筆者らの以前の研究[5]で用いた，Batch-axis Packing は，スループットを向上できる代わりに，CNN の各層において実行する準同型演算の数が多いため，レイテンシに関して効率的ではない．

4. 提案手法

本研究では，推論レイテンシを短縮するために，推論対象画像のチャネルごとの暗号化(Channel-wise Packing)と，学習済み CNN モデルに対する Channel Pruning の適用を組み合わせることを検証する．なお，Channel-wise Packing を使用した推論自体は Dathathri ら[13]や Lou ら[14]によって行われており，本稿でも同様の手法を採用する．

4.1. Channel-wise Packing による Batch-axis Packing と比較した準同型演算数の削減

Batch-axis Packing と Channel-wise Packing の推論処理中に必要な準同型演算の数を比較する．以降，CNN における入力画像や特徴マップのサイズをチャネル数C，高さH，幅Wを用いて，$C \times H \times W$のように示す．

Batch-axis Packing は，推論対象の複数画像における同位置のピクセルを単一の暗号文に格納する．つまり，暗号文のもつスロット数をS個としたとき，最大でS枚の$C \times H \times W$の画像を$C \times H \times W$個の暗号文に格納することができる．対して，Channel-wise Packing では，推論対象画像をチャネルごとに暗号文に格納する．つまり，1 枚の$C \times H \times W$の画像（$H \times W \leq S$を想定）はC個の暗号文に格納される．したがって，Batch-axis Packing と比較して，Channel-wise Packing を用いることで入力画像や特徴マップを表す暗号文数は$1/HW$倍となる．図 4.1 と図 4.2 に，$1 \times 5 \times 5$の画像（$C = 1, H = 5, W = 5$）に対して，それぞれ Batch-axis Packing，Channel-wise Packing を適用した例を示す（暗号文のスロット数は 8,192としている）．

表 4.1 に Batch-axis Packing と Channel-wise Packing の CNN の各層における主要な準同型演算数の比較を示す（暗号文と平文の積を$HMulPlain$，暗号文同士の和を$HAdd$，暗号文の二乗を$HSquare$とする）．表 4.1 において，畳み込み層と average pooling 及び活性化関数における入力を$IC \times IH \times IW$，出力を$\mathrm{OC} \times \mathrm{OH} \times \mathrm{OW}$とし，カーネルの高さ及び幅をそれぞれ$KH$，$KW$と表している．ここで，average pooling においては$IC = OC$，活性化関数においては$IC = OC$，$IH = OH$，$IW = OW$である．また，全結合層における入力ユニット数をIU，出力ユニット数をOUと表し，スロット数はSとする．なお，畳み込み層や全結合層でのバイアスの準同型加算は実行回数が少なく，レイテンシへの影響が少ないためカウントせず，活性化関数はシンプルのため二乗関数とする．また，Batch Normalization に関しては畳み込み層に結合されるため，0 回である．表 4.1 より，Batch-axis Packing と比較して，Channel-wise Packing を用いることで，準同型乗算数を畳み込み層，average pooling 及び活性化関数において$\frac{1}{OH \times OW}$，全結合層では$\frac{1}{IU}$に削減できることがわかる．準同型乗算数を大幅に削減できる代わりに，Channel-wise Packing では$rotation$の計算コストがかかるが，準同型乗算数の削減による恩恵のほうが大きいと考えられる．

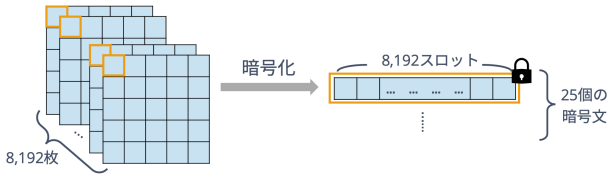

図 4.1 Batch-axis Packing の例

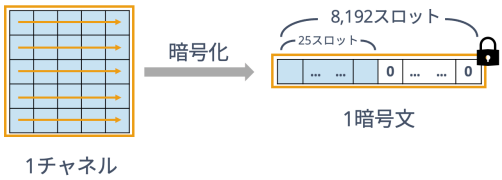

図 4.2 Channel-wise Packing の例

表 4.1 Batch-axis Packing と Channel-wise Packing の各層における準同型演算数の比較

	Batch-axis	Channel-wise
畳み込み層	$OC \times IC \times KH \times KW \times OH \times OW$回の $HMulPlain$及び$HAdd$	$OC \times IC \times KH \times KW$回の $HMulPlain$及び$HAdd$, $IC \times KH \times KW$回の $rotation$
average pooling	$IC \times KH \times KW \times OH \times OW$回の $HAdd$, $IC \times OH \times OW$回の $HMulPlain$	$IC \times KH \times KW$回の $HAdd$及び$rotation$, IC回の $HMulPlain$
活性化関数（二乗）	$IC \times OH \times OW$回の $HSquare$	IC回の $HSquare$
全結合層	$OU \times IU$回の $HMulPlain$及び$HAdd$	OU回の $HMulPlain$, $OU \times \log_2 S$回の $rotation$及び$HAdd$

4.2. Channel Pruning

深層学習において，モデルのパラメータ数やサイズを小さくするモデル圧縮手法の 1 つが Pruning（枝刈り）である．Channel Pruning は，Structured Pruning の一種であり，畳み込み層において，フィルタ単位で Pruning することで出力のチャネル数を削減する．本研究では Li ら[7]の手法を用いており，フィルタの L1 ノルム(重みパラメータの絶対値の総和)が小さいものから順に，指定した割合のフィルタを Pruning する．ある畳み込み層における Channel Pruning の例を図 4.3 に示す．図 4.3 のように，ある畳み込み層のフィルタをX%Pruning することにより，その畳み込み層の出力特徴マップのチャネル数と次の畳込み層のパラメータ数をX%削減することができる．つまり，表 4.1 における畳み込み層の出力チャネル数OCや，その畳み込み層に続く average pooling 及び活性化関数の入力チャネル数ICがX%減少する．したがって，Channel Pruning を適用することで，Batch-axis Packing 及び Channel-wise Packing どちらに対しても，Pruning した分だけ，準同型演算数を削減可能であることがわかる．

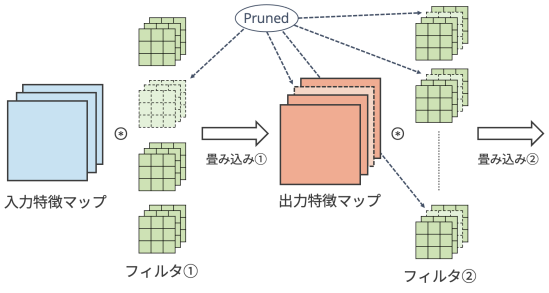

図 4.3 Channel Pruning の例

5. 評価実験

5.1. データセットとネットワーク構成

評価実験では，10 クラスの画像分類のタスクである MNIST データセット[15]と CIFAR-10 データセット[16]を用いた．

本実験で用いた CNN の構成は，筆者らの以前の研究[5]を元にした．ただし，CIFAR-10 データセットに対して適用する CNN は，1 つ目の全結合層のパラメータが全結合層のパラメータが全体の約 85%を占めていた（620,810 のうち 524,544）ため，1 つ目の全結合層を Global Average Pooling[17]に置換し，パラメータ数及び全結合層を削減した CNN に変更した．MNIST データセットに対して適用する CNN は，畳み込み層と全結合層の合計数が 3 であるため，3layer-CNN と呼ぶ．同様に，CIFAR-10 データセットに対して適用する 1 つ目の全結合層を Global Average Pooling に置換した CNN は，畳み込み層と全結合層の合計数が 4 であるため 4layer-CNN と呼ぶ．3layer-CNN の構成を表 5.1 に示し，4layer-CNN の構成を表 5.2 に示す．

表 5.1 MNIST データセットに用いる 3layer-CNN の構成 (総パラメータ数: 144,550)

層の種類	パラメータ	出力サイズ
Convolution	5×5のフィルタ5個 ストライド(2,2) パディングなし	5×12×12
Batch Normalization	—	5×12×12
Activation	—	5×12×12
Convolution	5×5のフィルタ50個 ストライド(2,2) パディングなし	50×4×4
Batch Normalization	—	50×4×4
Activation	—	50×4×4
Fully Connected	出力ユニット数 10	10

5.2. 平文での CNN モデルの学習および Channel Pruning

PyTorch フレームワーク[1]のバージョン 1.1.0 を用いて，5.1 項で示した CNN をそれぞれ MNIST データセットや CIFAR-10 データセットを用いて学習を行った．活性化関数は ReLU，Swish，二乗関数，表 5.3 に示した 2 次または 4 次の多項式で近似した ReLU 及び Swish の合計 11 通り用いた．表 5.3 の多項式近似活性化関数の名称において，rg は近似範囲の range を意味しており，deg は近似次数の degree を意味している．なお，活性化関数の多項式近似は python の scipy ライブラリ[2]で提供されている leastsq 関数[3]を使用し，近似範囲は

[1] https://pytorch.org/
[2] https://scipy.org/
[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.leastsq.html

筆者らの以前の研究[5]を元に，$[-5,5]$と$[-7,7]$の 2 通りを選択した．また，各データセットに対して学習に用いた設定を表 5.4 に示す．MNIST データセットと CIFAR-10 データセットのそれぞれに対して，学習により得られた各モデルを用いたテストデータに対する分類精度を表 5.5 と 5.6 に示す．また，最適化アルゴリズムは AdamW[18]とし，PyTorch におけるデフォルトパラメータ[4]($\gamma = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\varepsilon = 10^{-8}$，$\lambda = 0.01$)を使用した．

学習により得られたモデルに対し，Li ら[7]の手法を実装している Torch-Pruning[5]を用いて Channel Pruning を適用した．なお，Channel Pruning 及び準同型暗号上での推論の実験では，二乗関数と swish-rg5-deg2，swish-rg7-deg4 の 3 種類の活性化関数を使用したモデルのみを対象とした．二乗関数は，準同型暗号上での CNN 推論において頻繁に用いられるため採用した．また，swish-rg5-deg2 と swish-rg7-deg4 に関しては，表 5.5，5.6 において，ReLU 及び Swish の 2 次近似多項式と 4 次近似多項式の中でそれぞれ高い分類精度が得られた 2 つを採用した．3layer-CNN に対しては，2 つ目の畳み込み層のフィルタを 20%Pruning し，出力チャネル数を 50 から 40 に削減した．4layer-CNN に対しては，2 つ目の畳み込み層のフィルタを 10%，20%及び 30%の 3 通りで Pruning し，出力チャネル数をそれぞれ 64 から 58，52 及び 45 に削減した．それぞれのネットワークを 4layer-CNN-Pruned10%，4layer-CNN-Pruned20%，4layer-CNN-Pruned30%とする．ただし，二乗関数及び swish-rg5-deg2 を活性化関数として用いた場合は，2 つ目の畳み込み層のフィルタを 20%Pruning し再学習した段階で，分類精度が 10%以上低下したため，30%の Pruning は行わなかった．なお，Pruning 率は手動で設定した．

Channel Pruning を適用後，各モデルを 30 エポック再学習させた．MNIST データセットでは，3layer-CNN に対して，上記で設定した Channel Pruning と再学習のプロセスを 2 回行い，それぞれのネットワークを 3layer-CNN-Pruned-R1，3layer-CNN-Pruned-R2 とする（R は Round を意味している）．MNIST データセットにおける 3layer-CNN-Pruned-R1 及び R2 のテストデータに対する分類精度を表 5.7 に示す．CIFAR-10 データセットにおける 4layer-CNN-Pruned10%，20%，30%の 3 種類それぞれのテストデータに対する分類精度を表 5.8 に示す．

[4] https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

[5] https://github.com/VainF/Torch-Pruning

表 5.2 CIFAR-10 データセットに用いる 4layer-CNN の構成 (総パラメータ数: 94,986)

層の種類	パラメータ	出力サイズ
Convolution	3×3のフィルタ 32個 ストライド$(1,1)$ パディング$(1,1)$	$32\times32\times32$
Batch Normalization	—	$32\times32\times32$
Activation	—	$32\times32\times32$
Average Pooling	2×2のプールサイズ ストライド$(2,2)$	$32\times16\times16$
Convolution	3×3のフィルタ 64個 ストライド$(1,1)$ パディング$(1,1)$	$64\times16\times16$
Batch Normalization	—	$64\times16\times16$
Activation	—	$64\times16\times16$
Average Pooling	2×2のプールサイズ ストライド$(2,2)$	$64\times8\times8$
Convolution	3×3のフィルタ 128個 ストライド$(1,1)$ パディング$(1,1)$	$128\times8\times8$
Batch Normalization	—	$128\times8\times8$
Activation	—	$128\times8\times8$
Average Pooling	2×2のプールサイズ ストライド$(2,2)$	$128\times4\times4$
Fully Connected	出力ユニット数256	256
Fully Connected	出力ユニット数10	10

表 5.3 ReLU 及び Swish の多項式近似結果

活性化関数	近似次数	近似対象のx軸の範囲	名称	多項式近似結果
ReLU	2	$x\in[-5,5]$	relu-rg5-deg2	$0.09x^2+0.5x+0.47$
		$x\in[-7,7]$	relu-rg7-deg2	$0.0669x^2+0.5x+0.6569$
	4	$x\in[-5,5]$	relu-rg5-deg4	$-0.00327x^4+0.1639x^2+0.5x+0.2932$
		$x\in[-7,7]$	relu-rg7-deg4	$-0.00119x^4+0.11707x^2+0.5x+0.41056$
Swish	2	$x\in[-5,5]$	swish-rg5-deg2	$0.1x^2+0.5x+0.24$
		$x\in[-7,7]$	swish-rg7-deg2	$0.0723x^2+0.5x+0.4517$
	4	$x\in[-5,5]$	swish-rg5-deg4	$-0.00315x^4+0.17x^2+0.5x+0.07066$
		$x\in[-7,7]$	swish-rg7-deg4	$-0.001328x^4+0.128x^2+0.5x+0.1773$

表 5.4 MNIST 及び CIFAR-10 に対する学習の設定

	エポック数	バッチサイズ	データ拡張
MNIST	200	128	• ±15°の範囲でランダムに回転 • 0.8～1.2 倍の範囲でランダムに縮小/拡大
CIFAR-10	200	128	• 上下左右に 4 ピクセルのマージンを付与し，ランダムに32×32の領域を切り出し • 1/2の確率で水平方向に反転

表 5.5 MNIST に対する平文での分類精度

ネットワーク	活性化関数	分類精度 [%]
3layer-CNN	ReLU	99.40
	Swish	99.55
	二乗	99.35
	relu-rg5-deg2	99.36
	relu-rg7-deg2	99.49
	relu-rg5-deg4	99.39
	relu-rg7-deg4	99.43
	swish-rg5-deg2	99.48
	swish-rg7-deg2	99.42
	swish-rg5-deg4	99.48
	swish-rg7-deg4	99.52

表 5.6 CIFAR-10 に対する平文での分類精度

ネットワーク	活性化関数	分類精度 [%]
4layer-CNN	ReLU	85.71
	Swish	85.58
	二乗	79.17
	relu-rg5-deg2	78.79
	relu-rg7-deg2	78.29
	relu-rg5-deg4	79.48
	relu-rg7-deg4	79.95
	swish-rg5-deg2	79.53
	swish-rg7-deg2	78.40
	swish-rg5-deg4	80.78
	swish-rg7-deg4	80.91

表 5.7 MNIST データセットにおいて 3layer-CNN に Channel Pruning を適用したモデルの分類精度 (丸括弧（）内は Pruning 適用前の分類精度との差分)

ネットワーク	活性化関数	分類精度 [%]
3layer-CNN-Pruned-R1 パラメータ数: 11,670	二乗	98.61 (-0.74)
	swish-rg5-deg2	98.83 (-0.65)
	swish-rg7-deg4	97.77 (-1.75)
3layer-CNN-Pruned-R2 パラメータ数: 9,366	二乗	96.68 (-2.67)
	swish-rg5-deg2	95.52 (-3.96)
	swish-rg7-deg4	96.78 (-2.74)

表 5.8 CIFAR-10 データセットにおいて 4layer-CNN に Channel Pruning を適用したモデルの分類精度 (丸括弧（）内は Pruning 適用前の分類精度との差分)

ネットワーク	活性化関数	分類精度 [%]
4layer-CNN-Pruned10% パラメータ数: 86,328	二乗	72.73 (-6.44)
	swish-rg5-deg2	73.93 (-5.60)
	swish-rg7-deg4	76.69 (-4.22)
4layer-CNN-Pruned20% パラメータ数: 77,670	二乗	66.70 (-12.47)
	swish-rg5-deg2	64.94 (-14.59)
	swish-rg7-deg4	73.18 (-7.73)
4layer-CNN-Pruned30% パラメータ数: 67,569	swish-rg7-deg4	65.20 (-15.71)

5.3. 評価方法

CKKS 方式を実装している準同型暗号ライブラリとして SEAL (バージョン 3.6.6)[19]を用いて準同型暗号上での CNN 推論処理を実装した．なお，スループット重視の Batch-axis Packing で暗号化した暗号文上での推論と，レイテンシ重視の Channel-wise Packing で暗号化した暗号文上での推論の 2 パターンを実装し，準同型演算数及び推論レイテンシを比較する．なお，準同型演算数はレイテンシへの影響が大きいと予想される準同型乗算(暗号文と平文の積を計算する *HMulPlain*，暗号文の二乗を計算する *HSquare* の 2 項目)と *rotation*，*relinearization*，*rescaling* の計 5 項目を計測し，報告する．Batch-axis Packing での推論レイテンシは，同一プロセス内で連続した 4 回の推論処理を実行し，1 回目を除いた 3 回の平均値を報告する．Channel-wise Packing での推論レイテンシは，10,000 回の推論レイテンシの平均値を報告する．

CNN は，5.2 項において学習した 2 種類のモデル(3layer-CNN，4layer-CNN)と各モデルに対し Channel Pruning を適用したモデルを対象とし，活性化関数は 5.2 項で述べたように，二乗関数と swish-rg5-deg2，swish-rg7-deg4 の 3 種類を用いた．また，筆者らの以前の研究 [5] と同様に，畳み込み層と Batch Normalization の結合や多項式近似活性化関数及び average pooling の係数操作といった最適化を適用し，暗号文の初期レベル L を可能な限り小さくした．加えて，CKKS 方式は固定小数点数で格納する値を表現するため，推論処理の途中で暗号文に格納されている全ての値が 0 と等しくなり，実質暗号文として扱えず処理が続行できないケースがある．したがって，重みパラメータ(最適化による畳み込み層と Batch Normalization の結合時の重みや，その値に多項式近似活性化関数の最高次係数や average pooling の係数をかけたものを含む)の絶対値が閾値 ε 未満の場合，値を丸めることで回避するようにした．具体的には重みパラメータを w として，$0.0 \leq w < \varepsilon$ のとき $w = \varepsilon$，$-\varepsilon < w < 0.0$ のとき $w = -\varepsilon$ とした．閾値 ε は，対象モデルが 3layer-CNN かつ活性化関数が swish-rg7-deg4 の場合 $\varepsilon = 10^{-6}$，それ以外の場合は $\varepsilon = 10^{-7}$ と設定した．

本実験を行った計算機の CPU は Intel Xeon E7-8880 v3(2.30GHz)であり，1CPU あたりの物理コア数は 18 コアである．また，CPU 数は 4 で，メモリは 3TB である．なお，OpenMP[6](バージョン 4.5)によるスレッド並列を適用し，使用可能なコアを全て活用するため，使用スレッド数は 72 に設定し，測定を行った．

5.4. 評価結果

5.4.1. MNIST データセットでの測定結果

MNIST データセットに対する準同型暗号上での推論処理の測定結果を表 5.9 に示す．なお，分類精度の列の丸括弧内の数値は，Batch-axis Packing を用いた暗号文での分類精度である．

表 5.9 より，Channel-wise Packing は Batch-axis Packing と比較して，*rotation* 以外の準同型演算の実行回数を削減できており，平均 89%レイテンシが短い．また，Channel Pruning により，Batch-axis Packing，Channel-wise Packing での推論における準同型演算数を削減することができ，レイテンシの短縮が達成でき

[6] https://www.openmp.org/

た．Batch-axis Packing では，活性化関数として swish-rg5-deg2 を用いた 3layer-CNN-Pruned-R2 の場合，最短のレイテンシであり，4.71 秒であった．対して，Channel-wise Packing における最短のレイテンシは，活性化関数として二乗関数または swish-rg5-deg2 を用いた 3layer-CNN-Pruned-R2 の場合の 0.60 秒であり，Batch-axis Packing での最短レイテンシと比較すると，約 1/8 倍であった．そして，最も高い分類精度(99.52%)が得られた，活性化関数として swish-rg7-deg4 を用いた 3layer-CNN では，0.97 秒の推論レイテンシを達成した．

表 5.9 MNIST データセットに対する測定結果

		ネットワーク					
		3layer-CNN		3layer-CNN-Pruned-R1		3layer-CNN-Pruned-R2	
二乗	分類精度(平文) [%]	99.35		98.61		96.68	
	パッキング	Batch	Channel	Batch	Channel	Batch	Channel
	分類精度(暗号文) [%]	99.35	99.35	98.54	98.54	96.58	96.58
	# of *HMulPlain* ($\times 10^3$)	126.0	6.4	104.4	5.1	87.1	4.1
	# of *HSquare*	1,520	55	1,360	45	1,232	37
	# of *rotation*	0	330	0	320	0	312
	# of *relin.*	1,520	55	1,360	45	1,232	37
	# of *resc.*	3,050	120	2,730	100	2,474	84
	レイテンシ [sec]	6.44	0.71	5.99	0.65	5.37	0.60
swish-rg5-deg2	分類精度(平文) [%]	99.48		98.83		95.52	
	パッキング	Batch	Channel	Batch	Channel	Batch	Channel
	分類精度(暗号文) [%]	99.48	99.48	98.80	98.80	95.32	95.32
	# of *HMulPlain* ($\times 10^3$)	127.5	6.4	105.8	5.2	88.4	4.2
	# of *HSquare*	1,520	55	1,360	45	1,232	37
	# of *rotation*	0	330	0	320	0	312
	# of *relin.*	1,520	55	1,360	45	1,232	37
	# of *resc.*	3,050	120	2,730	100	2,474	84
	レイテンシ [sec]	6.35	0.72	5.25	0.65	4.71	0.60
swish-rg7-deg4	分類精度(平文) [%]	99.52		97.77		96.78	
	パッキング	Batch	Channel	Batch	Channel	Batch	Channel
	分類精度(暗号文) [%]	99.52	99.52	97.63	97.63	96.62	96.62
	# of *HMulPlain* ($\times 10^3$)	129.0	6.5	107.1	5.2	89.6	4.2
	# of *HSquare*	3,040	110	2,720	90	2,464	74
	# of *rotation*	0	330	0	320	0	312
	# of *relin.*	3,040	110	2,720	90	2,464	74
	# of *resc.*	4,570	174	4,088	145	3,706	121
	レイテンシ [sec]	10.72	0.97	9.94	0.88	8.21	0.84

5.4.2. 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準同型暗号上での推論処理の測定結果を表 5.10 に示す．表 5.10 より，MNIST での測定結果と同様に，Channel-wise Packing は Batch-axis Packing と比較して，準同型演算の実行回数を削減できており，平均 98%のレイテンシ短縮を達成した．Batch-axis Packing における最短のレイテンシは，Channel Pruning を適用した 4layer-CNN-Pruned20%において，活性化関数として二乗関数を用いた場合であり，447.1 秒であった．同様に，Channel-wise Packing における最短のレイテンシも，Channel Pruning を適用した 4layer-CNN-Pruned20%において，活性化関数として二乗関数を用いた場合であり，8.1 秒であった．これは，Batch-axis Packing の最短レイテンシ 447.1 秒と比較し，約 1/55 倍である．また，最も高い分類精度(80.96%)が得られた，活性化関数として swish-rg7-deg4 を用いた 4layer-CNN では，12.9 秒の推論レイテンシを達成した．

CIFAR-10 データセットでは，MNIST データセットでの実験結果と比べて Channel Pruning による分類精度の低下が大きかったが，Batch-axis Packing，Channel-wise Packing ともに準同型演算数の減少により，レイテンシの短縮を達成することができた．3 通りの Pruning 率(2 つ目の畳み込み層のフィルタに対して 10%，20%及び 30%)を適用した，活性化関数として swish-rg7-deg4 を使用した場合の Channel-wise Packing における分類精度と推論レイテンシの関係を示したグラフを図 5.1 に示す．図 5.1 より，Pruning 率が高い(より多くのパラメータを削除する)ほど，分類精度は低下するが，推論レイテンシが短縮できていること(分類精度と推論レイテンシのトレードオフ)が確認できる．また図 5.1 において，Pruning 前のモデルである 4layer-CNN と，2 つ目の畳み込み層のフィルタを 20%Pruning した 4layer-CNN-Pruned20%を比較すると，約 8%の分類精度の低下に対し，約 19%のレイテンシの短縮を達成した．したがって，準同型暗号上での CNN 推論において，事前に学習済みモデルに対し Channel Pruning を適用することにより，多少の分類精度の低下を許容する代わりに，推論レイテンシを短縮可能であるといえる．

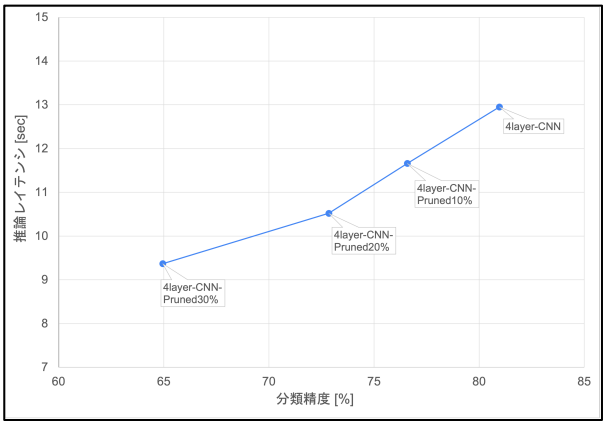

図 5.1 CIFAR-10 データセットでの Channel-wise Packing を用いた推論における Channel Pruning による分類精度と推論レイテンシの関係 (活性化関数: swish-rg7-deg4)

表 5.10 CIFAR-10 データセットに対する測定結果

		ネットワーク					
		4layer-CNN		4layer-CNN-Pruned10%		4layer-CNN-Pruned20%	
二乗	分類精度(平文) [%]	79.17		72.73		66.70	
	パッキング	Batch	Channel	Batch	Channel	Batch	Channel
	分類精度(暗号文) [%]	79.22	79.22	72.14	72.42	66.27	66.52
	# of *HMulPlain* ($\times 10^3$)	9,148	93.2	8,370	84.5	7,592	75.9
	# of *HSquare*	57.3	0.2	55.8	0.2	54.3	0.2
	# of *rotation*	0	4.1	0	4.0	0	3.9
	# of *relin.*	57.3	0.2	55.8	0.2	54.3	0.2
	# of *resc.*	114.7	0.6	111.6	0.6	108.5	0.6
	レイテンシ [sec]	505.6	9.8	490.5	8.9	447.1	8.1
swish-rg5-deg2	分類精度(平文) [%]	79.53		73.93		64.94	
	パッキング	Batch	Channel	Batch	Channel	Batch	Channel
	分類精度(暗号文) [%]	79.52	79.55	73.84	73.79	64.40	64.35
	# of *HMulPlain* ($\times 10^3$)	9,205	93.4	8,426	84.7	7,646	76.1
	# of *HSquare*	57.3	0.2	55.8	0.2	54.2	0.2
	# of *rotation*	0	4.1	0	4.0	0	3.9
	# of *relin.*	57.3	0.2	55.7	0.2	54.2	0.2
	# of *resc.*	114.7	0.6	111.3	0.6	108.2	0.6
	レイテンシ [sec]	526.7	9.6	488.1	8.8	456.5	8.2
swish-rg7-deg4	分類精度(平文) [%]	80.91		76.69		73.18	
	パッキング	Batch	Channel	Batch	Channel	Batch	Channel
	分類精度(暗号文) [%]	80.72	80.96	76.34	76.58	72.76	72.85
	# of *HMulPlain* ($\times 10^3$)	9,263	93.6	8,482	85.0	7,701	76.3
	# of *HSquare*	114.7	0.4	111.6	0.4	108.5	0.4
	# of *rotation*	0	4.1	0	4.0	0	3.9
	# of *relin.*	114.7	0.4	111.6	0.4	108.5	0.4
	# of *resc.*	171.4	0.8	166.5	0.8	166.5	0.8
	レイテンシ [sec]	851.3	12.9	786.3	11.7	727.2	10.5

6. まとめ

本稿では，準同型暗号上での CNN 推論処理において，レイテンシの短縮を目指した．具体的には，推論対象データのチャネルごとの暗号化(Channel-wise Packing)を適用し，さらに CNN モデルに対し事前に Channel Pruning を適用することで，準同型演算の回数を削減した．

評価実験では，推論対象の複数画像をピクセルごとに暗号化し推論を行う Batch-axis Packing を比較対象とし，MNIST データセット及び CIFAR-10 データセットを用いて，推論レイテンシの測定を行った．MNIST データセットでは，Channel-wise Packing により Batch-axis Packing と比較して，レイテンシを平均 89% 短縮することができた．また Channel Pruning を適用した場合，最短 0.60 秒の推論レイテンシを達成した．CIFAR-10 データセットでは，Channel-wise Packing により Batch-axis Packing と比較して，レイテンシを平均 98%短縮することができた．また Channel Pruning を適用した場合，最短 8.1 秒の推論レイテンシを達成した．

今後の展望の 1 つとして，VGG や ResNet といったより大規模なモデルに対し Pruning を適用し，準同型暗号上で推論処理を評価できるようにすることが挙げられる．

謝　辞

本研究は，JST CREST（JPMJCR1503）の支援を受けたものである．

参　考　文　献

[1] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pp. 169-178, 2009.

[2] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers," in Advances in Cryptology - ASIACRYPT 2017, pp. 409-437, 2017.

[3] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A Full RNS Variant of Approximate Homomorphic Encryption," in Selected Areas in Cryptography - SAC 2018, pp. 347-368, 2019.

[4] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," Designs, Codes and Cryptography, vol. 71, no. 1, pp. 57-81, 2014.

[5] T. Ishiyama, T. Suzuki, and H. Yamana, "Highly accurate CNN inference using approximate activation functions over homomorphic encryption," in 2020 IEEE international conference on big data (big data), pp. 3989-3995, 2020.

[6] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in 2017 IEEE international conference on computer vision (ICCV), pp. 1398-1406, 2017.

[7] H. Li, H. S. Asim Kadav, Igor Durdanovic, and H. P. Graf, "Pruning filters for efficient ConvNets," in 5th International Conference on Learning Representations, 2017.

[8] 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 *Proceedings of the 33rd international conference on international conference on machine learning - volume 48*, pp. 201–210, 2016.

[9] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, “Privacy-preserving classification on deep neural network,” In *IACR Cryptology ePrint Archive, Report 2017/035*, 2017.

[10] A. Al Badawi *et al.*, “Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs,” *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1330–1343, 2021.

[11] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, “NGraph-HE: A graph compiler for deep learning on homomorphically encrypted data,” in *Proceedings of the 16th ACM international conference on computing frontiers*, pp. 3-13, 2019.

[12] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, “Faster CryptoNets: Leveraging sparsity for real-world encrypted inference,” In *arXiv preprint arXiv:1811.09953*, 2018.

[13] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, “CHET: An optimizing compiler for fully-homomorphic neural-network inferencing,” in *Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation*, pp. 142–156, 2019.

[14] Q. Lou and L. Jiang, “HEMET: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture,” in *Proceedings of the 38th international conference on machine learning*, vol. 139, pp. 7102–7110, 2021.

[15] Y. LeCun, C. Cortes, and C. J. C. Burges, "MNIST handwritten digit database." http://yann.lecun.com/exdb/mnist/, accessed Jan. 7. 2021.

[16] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian Institute for Advanced Research)." http://www.cs.toronto.edu/~kriz/cifar.html, accessed Jan. 7. 2021.

[17] M. Lin, Q. Chen, and S. Yan, “Network in network,” in *arXiv Preprint arXiv:1312.4400*, 2014.

[18] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *arXiv Preprint arXiv: 1711.05101*, 2019.

[19] “Microsoft SEAL (release 3.6).” https://github.com/Microsoft/SEAL, 2020.