Vol.2018-ARC-229 No.22
Vol.2018-SLDM-182 No.22
2018/1/19

DNN 向けニューロン毎の量子化ビット幅最適化に関する評価

石井 潤[1,a)]　坂本 龍一[1]　近藤 正章[1]

概要：
近年，Deep Neural Network (DNN) において，重みや入力値のビット幅を削減して記憶領域と計算資源のコストを削減する研究が多く行われている．我々は，ネットワークの各層および層内のニューロン毎にビット幅を最適化する手法とそのアーキテクチャを検討している．本稿では，ニューロン毎の量子化を適用したネットワークに再学習を行った場合の評価や，量子化をモデル全体に適用する評価を行った．再学習を行わない場合と比較して 1.82 倍パラメータサイズを削減できることがわかった．

1. はじめに

深層ニューラルネットワーク (Deep Neural Network: DNN）は，多層のニューラルネットワークを用いた機械学習モデルであり，コンピュータビジョン [7]，音声認識や自然言語処理 [10] などの様々な分野で高い認識性能が報告されている．近年の DNN モデルは，精度を向上させるためにネットワークモデルが大規模になっており，必要な計算量やメモリ消費量が増加する傾向にある．この計算量およびメモリアクセス量の増大は，消費エネルギーの増加を招く．計算資源や電力資源が限られる組み込み機器において，大規模な CNN を実行する際にはその演算およびメモリアクセスに要する消費エネルギーの削減と，高い性能を両立することが課題となる．

これまで，組み込み機器向けの低消費電力な DNN 専用アクセラレータが多く提案されてきた．DaDianNao[1] は，チップ内に大容量の eDRAM を配置することで，オフチップアクセス回数を削減し，処理の高速化と低消費電力化を実現したアクセラレータである．Eyeriss[2] は，計算量の多い畳み込み層を高電力効率で処理することを狙ったアクセラレータである．演算ユニットを二次元アレイ状に配置し，データの再利用性や並列性を考慮してデータフローを最適化することで，畳み込み層での計算効率を高めるとともにオフチップメモリアクセスを削減し，高速化と低消費電力化を実現する．EIE[5] は，パラメータ数が多い全結合層におけるメモリアクセスのボトルネックを解決し，高電力効率を狙うアクセラレータである．圧縮したネットワークモデル [4] を使用することでメモリアクセス負荷を低減し，また圧縮に伴う複雑なデータ構造の処理を効率的に行うハードウェア機構を持つことで，CPU や GPU などの汎用ハードウェアよりも高速，低消費電力な処理を実現可能である．

一方で，学習済みネットワークを圧縮することにより，演算負荷や消費電力を削減する手法も多数提案されている．枝刈り [3] はシナプスやニューロンの重要度をもとに，重要度の低いものを削除することでネットワークモデルのパラメータ数を削減する手法であり，認識精度を落とさずにモデルサイズを 10%程度まで圧縮することに成功している．前述の EIE では，枝刈りにより削除されたパラメータを 0 で表現し，Zero skipping によって計算を省略することで計算回数を低減している [5].

量子化 [15] は，単精度浮動小数点数などで表現された値を，より低精度な数値表現で近似することで，計算負荷やメモリ消費量の削減を行う手法である．DNN の計算では，推論時に必要な認識精度を得るための各データにはそれほど高い精度が必要でないことがわかっている．そこで，入力や重みの量子化ビット幅を削減することで，認識精度の低下を抑えながらデータあたりの計算に必要なハードウェア面積やメモリ消費量，あるいはメモリアクセスのデータ転送量を削減することができる．これまで量子化手法として，ネットワークモデル全体で同一の量子化ビット幅を用いるもの [15]，畳み込み層や全結合層などの層の種類毎に個別の量子化ビット幅を用いるもの [4]，さらには，各層で異なるビット幅で量子化を行うことで，認識精度を低下させずにさらにメモリ消費量を削減するもの [12] などが提案されている．また，各ビットを 2 のべき乗に対応する値として非線形に量子化を行う手法も提案されている [11][9].

我々は，さらに計算負荷やデータ転送量を削減するために，ニューロン毎に異なる量子化ビット幅を適用することを検討している [20]．ニューロンに接続されるシナプスの

[1] 東京大学 大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[a)] ishii@hal.ipc.i.u-tokyo.ac.jp

重みの値のレンジは，各ニューロン毎に大きく異なるため，重みの値の分散が大きいニューロンの重みには多数のビット幅を，重みの値の分散が小さいニューロンの重みには少数のビットを割り当てるよう量子化することで，従来の量子化の手法に比べてさらにデータ量を圧縮できる (あるいは同じデータ量で精度を向上させることができる) と期待される．我々の予備評価において，3 層の単純な全結合ネットワークに対し，MNIST データセット [8] を用いて評価を行った結果，層毎の量子化に比べてニューロン毎の量子化を行うことで，ほぼ同等の予測精度を保ちつつデータをさらに圧縮できる可能性があることを示している．

本稿では，ニューロン毎の量子化を適用したネットワークに再学習を行った場合の評価や，量子化をモデル全体に適用する評価を行い，本手法の有効性を定量的に考察する．

2. 量子化手法と関連研究

2.1 量子化手法

DNN における量子化とは，浮動小数点数から固定小数点数への変換や，1 データのビット幅を縮小させることで，ハードウェアコストやメモリ消費量を低減する手法である．量子化にともない，表現可能なデータの範囲が縮小するため，データの精度が低下し，それにより計算精度が低下する．DNN は一般に計算精度を低下させても認識精度が低下しにくいことが知られている．DNN による分類問題は，認識対象が各クラスに属する確率を計算し，その順位に基いて分類の予測がなされることが通常である．したがって，確率同士の大小関係が重要であり絶対値は重要でないため，単精度浮動小数点数ほどの高い計算精度は必要ない [15]．したがって，認識精度の低下を 0%から 10 数%程度に抑えながら，ビット幅を例えば 32 ビットから 16 ビット以下のより小さいビット幅に低減させる手法が多く提案されている．

DNN の推論計算時において，量子化の対象は重みのみの場合もあれば，重みと入力値の両方である場合もある．前者はメモリ消費量の削減が主な効果で，計算時には一般的にデータがデコードされ，フル精度で計算が行われる．後者は積和演算における 2 つの入力の双方が量子化されるため，メモリ消費量の削減だけでなく，演算器をコストの小さな演算器に置き換えることができる．

2.1.1 線形な量子化

量子化手法は，浮動小数点数を固定小数点数で近似したり，入力や重みデータの最大値に合わせてスケーリングを行い小数点の位置を決めた後，下位のビットを削減することで行われることが一般的である．

DNN の量子化はモデル全体あるいは層の種類ごとに一律のビット幅が用いられることが多い．画像認識コンペティション ILSVRC2012 の勝者である AlexNet という CNN モデルにおいて，認識精度の劣化なしで，モデル全体で入力と重みの両方を 9 ビットに量子化できることが報告されている [12]．また，層毎にビット幅の異なった量子化をすることで，モデル全体を量子化したときのビット幅よりも大きくビット幅を削減する手法が提案されている [12]．各層の入力や重みの値の分散の違いなどから，層毎で最適なビット幅が異なると考えられている．

また，Binarized Neural Network (BNN) で知られるように，学習した重みを 2 値，3 値で再学習して，メモリ消費量とハードウェアコストを大幅に削減する手法も報告されている [6], [13], [16]．2 値化ネットワーク [13] は，認識精度の低下を AlexNet で 11 ポイント程度に抑えつつ，入力と重みの双方を 2 値化している．重みを 2 値にすることでメモリ消費量を 1/32 に削減し，積和演算の積を XNOR，和を Population Count というビットレベルの極めて単純な演算に帰着させることで，処理を高速・低消費電力に実行できる．3 値化ネットワーク [16] は，認識精度の低下を 1%以内に抑えながら，重みを-1, 0, +1 の 3 値，すなわち 2 ビットで表現することで，重みの容量を 1/16 に削減する．ハードウェアで実行する際，重みが 0 のときは計算をスキップし，0 以外のときは符号のみに乗算を適用することで処理を高速化できる．

2.1.2 非線形な量子化

前節で述べた量子化手法は，刻み幅が一律であるが，一律でない刻み幅を採用することで，少ないビット長で大きな範囲のデータを表現することができるため，ビット幅をさらに削減する手法が提案されている．

Weight-Sharing[4] は，重みを K-means クラスタリングによって数種類に分類する量子化手法である．AlexNet において，認識精度の低下なしに畳み込み層で 8 ビット，全結合層で 4 ビットまでビット幅を削減できると報告されている．この手法は線形な量子化よりもさらにビット幅を削減できる一方，重みの値をインデックスから真値にデコードする計算が必要になり，オーバーヘッドが大きいという欠点がある．

対数量子化手法 [9] は，量子化の刻み幅に対数を用いた手法である．底に 2 を用いた量子化をすることで，値が 2 の冪になるため，乗算ビットシフトに帰着させることでハードウェアコストを削減できる．AlexNet を対数量子化した実験によると，認識精度の低下は 3.7 ポイントで，重みをと入力を 5 ビットまで削減できると報告されている．

3. ニューロン毎の量子化手法

3.1 最適なビット幅の探索手法

本節では，文献 [20] で提案したニューロン毎に異なる量子化を行う手法の概要を述べる．

本手法は，まず層毎に量子化を行い最適なビット幅を設定し，その後ニューロン毎の量子化を行うというステップからなる．層毎の最適なビット幅の探索は，[12] の手法に従い，第 1 層から順番に th を満たす最小のビットを求めていく．また，ニューロン毎の最適化は本稿では全結合層に限定して行うものとする．

量子化により認識精度は低下する可能性があるが，許容できる認識精度の低下割合の最小値を th とする．例えば，$th = 0.99$ のときは完全精度の 99%までの認識精度低下を

許容することを示す．以下に，ある層におけるビット幅最適化の手順を示す．

(1) 重みのビット幅を 16 ビット (初期値) にする
(2) 推論計算を行った場合の認識精度を求める
(3) 重みパラメータのビット幅を 1 小さくし認識精度を求める
(4) 認識精度が th を満たす場合は，再び (3) から処理を行い，満たさない場合は (5) に移る
(5) 現在のビット幅に 1 を加えたものを当該層の最適なビット幅とする
(6) この層で求めた最適なビット幅を利用し，次層以降でも同様の処理を (1) から行う

この手法では，最適なビット幅の組み合わせが求まるとは限らない．ただし，上層から伝播される情報による精度の劣化を防ぎながら下層へと探索を行っていくため妥当な手法であると考えられる．

次に，ニューロン毎のビット幅の最適化について述べる．まず，上記で述べた層毎の最適なビット幅が求まっていると仮定する．ニューロン毎にも層毎のビット幅最適化の場合と同様に，th を満たす範囲で徐々にビット幅を縮小させるという貪欲法的な手法を用いる．

まず，各層において，入力される重みのビット幅を縮小するニューロンの割合を r と定める．ある層について，th を満たす最大の r を求める．この際，ビット幅を縮小するニューロンの選択方法は，本稿では以下に示す単純な方法を検討する．

対象の問題のテストデータセットに対する推論計算において，ニューロン毎の出力を記憶する．完全精度の場合と量子化後の精度での差が大きいニューロンは，そのニューロンに入力される重みの精度が不足しているものと考え，大きいビット幅を与える．それ以外は小さいビット幅を与える．本稿では，大きいビット幅を n_l ビットとし，そのデータを長ビットデータと呼ぶ．また，小さいビット幅を n_s ビットとし，そのデータを短ビットデータと呼ぶ．先に述べたように，各層でそれぞれの割合は $1-r:r$ となる．以下にビット幅を短ビットにするニューロンを選択する手順をまとめる．

(1) テストデータに対して完全精度で推論処理を行い各ニューロンの出力値を記憶する
(2) 層毎の量子化で求めたビット幅で推論処理を行い各ニューロンの出力値を記憶する
(3) ニューロン毎に完全精度と量子化後の各出力値の差を求め，それを加算し，平均を記憶する
(4) $r = 0.1$ とする
(5) 差の平均が下位 r の割合のニューロンを接続する重みをさらに縮小できるニューロンとして選択する
(6) ニューロン毎に量子化した場合での認識精度を求める
(7) 認識精度が th を満たす場合は，r の値を増加させて再度 (5) から繰り返す
(8) 認識精度が th を満たさない場合は，先の r の値で選択されたニューロン毎の量子化を採用する

3.2 3 層ニューラルネットワークの例

本節では，3 層ニューラルネットワークでのニューロン毎の量子化の具体例を示す．なお，以降では第 1 層から第 3 層までのビット幅を (n_{l1}, n_{l2}, n_{l3}) と表記して説明する．

まず，学習済みの 3 層のニューラルネットワークを用意する．このとき，層ごとのビット幅は (32, 32, 32) である．次に，層ごとの量子化を行うと，例えば重みのビット幅は (6, 4, 4) になる，

次に，第 1 層についてニューロン毎の量子化を行う．誤差が小さい方から割合 r のニューロンの重みを半分の精度である 3 ビットにする．$r = 0.1, 0.2, 0.3...$ と r の値を大きくしていき，例えば $th = 0.99$，つまり認識精度がベースラインの 99%を上回る最大の r を探索する．例えば $r = 0.1$ のとき，重みの 90%が 6 ビット，10%が 3 ビットとなり，認識精度がベースラインの 99%を上回っていたとすると，r の値を 0.2 にして再度ニューロン毎の量子化を行い，認識精度を求める．これを繰り返すと，例えば r の値が 0.7 になったとき，認識精度がベースラインの 99%を下回ったとすると，最適な割合 r は 0.6 となる．

最適な割合 $r = 0.6$ が見つかったら，その層に対して 0.4:0.6 の割合で長ビットデータは 6 ビット，短ビットデータは 3 ビットの量子化を行う．当該層での割合 r と量子化するニューロンが求められたら，次の層で同様の計算を繰り返す．

3.3 再学習

本稿ではニューロン毎の量子化後に再学習を行うことで，量子化による認識精度の低下を補償することを検討する．これにより，同等の認識精度を保持しつつ，さらにビット幅を削減できる可能性がある．

再学習には，[17] および [18] で用いられた技術を使用する．ネットワークモデルは完全精度の重み w をもち，量子化した重み w' は w から得る．推論計算の際には，量子化後の重み w' を用いて計算を行う．一方学習では，完全精度の重み w を更新する．w' を用いて推論計算して求めた誤差を逆伝播し，重みの更新 Δw を w に適用する．

4. ニューロン毎の量子化のためのアーキテクチャ拡張

4.1 ベースのハードウェア構成

我々は，高い消費電力効率と様々なネットワークモデルを実行可能な DNN 推論用のアクセラレータを開発している [19]．図 1 に示すように，マイクロコントローラと SIMD 型積和演算器からなるコアを複数搭載したマルチコアアクセラレータである．

コアは回路規模の小さなマイクロコントローラと SIMD 型積和演算器から構成される．マイクロコントローラは 16 ビット固定長命令を用いることで，小型で高電力効率となっている．さらに，積和演算を効率よく実行するために，SIMD 型積和演算器と独自のカスタム SIMD 算術命令を実装している．

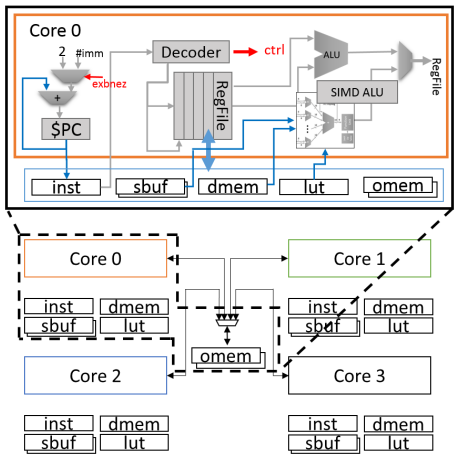

図 1 4 コア構成のアクセラレータ

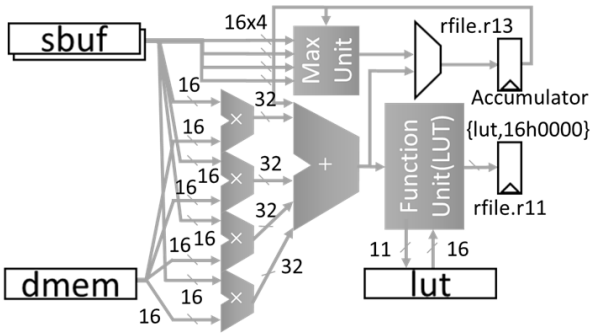

図 2 SIMD 型積和演算器

メモリ構成

各コアが持つ 5 つのメモリは inst, sbuf, dmem, lut, omem からなり，それぞれがアドレス空間に割り付けられている．inst は命令メモリで，sbuf と dmem は処理対象データ用メモリである．sbuf は再利用性の低いストリームデータ用のバッファとして，dmem は再利用性の高いデータ用のバッファとして用いる．sbuf 側はデータ転送量が多く，メモリボトルネックの原因となりやすいため，sbuf 側のみダブルバッファリングを行うことで実行時間の削減を狙う．lut は活性化関数に利用し，omem は出力データ用である．

畳込み層では学習済み重みパラメータの再利用性が高く各層の入力データの再利用性が低いが，全結合層では逆転し各層の入力データの再利用性が高く学習済み重みパラメータの再利用性が低いという違いがあり，これらの特性の違いに対応可能である．

マイクロコントローラ

本アーキテクチャは 4 段パイプラインのインオーダー実行で，MIPS に近い形式の 16 ビット固定長命令セットを解釈実行する．マイクロコントローラは SIMD 型積和演算器の制御やループ制御，メモリへの load/store を制御する．レジスタファイルは 32 ビット 16 本であるが，そのうち 4 本を SIMD 型積和演算器の演算結果が格納される特殊レジスタとして割り当てている．

SIMD 型積和演算器

SIMD 型積和演算器の基本構成を図 2 に示す．SIMD 型積和演算器は 16 ビット長データ 4 並列で演算を行うことができ，実行可能な演算は積和演算と MAX 演算である．処理対象データは sbuf と dmem から直接演算器に供給され，データバスは 64 ビット幅である．lut は活性化関数に利用する．先行研究では活性化関数に ReLU 関数のみをサポートするものもあるが，汎用性の観点からルックアップテーブルによる実装を採用している．演算結果は，直接レジスタに保存される．

sbuf と dmem それぞれから 16 ビット固定小数点形式サイズ 4 のベクトルデータが SIMD 型積和演算器に投入され，その内積演算結果がレジスタにアキュムレートされる．前述のマルチサイクルのカスタム SIMD 算術命令は，積和演算や MAX 演算を sbuf と dmem アドレスをインクリメントしながら指定回数連続実行する．

4.2 アーキテクチャの拡張

層毎に量子化ビット幅を変更するような粗粒度な最適化の場合は，実行前にハードウェアのモードを変更することで，実現することが可能である．一方で，ニューロン毎に重みのデータの幅を変えるような細粒度な最適化を行う場合には，重みのデータをチップ内のバッファに転送する際に，ビット幅の異なるデータが混在する可能性もあり，各データのビット幅の判定を効率良く行うことが重要となる．そのためのアーキテクチャを，前節で述べたアクセラレータを拡張することで実現する．なお，本拡張はアクセラレータアーキテクチャとは独立であるため，他のアーキテクチャにも適用可能であると考えられる．

図 3 に拡張部分のハードウェア構成を示す．これは，図 2 の SIMD 演算器部分に相当する．まず，sbuf の 1 エントリのビット幅は半分になり，各演算器へと供給するビット幅も半分になる．また，sbuf には重みデータを保存するため，本メモリを Half Weight Memory と呼ぶこととする．また，Half Weight Memory の各エントリへのフラグビットを示す Double flag と呼ぶ領域を新たに sbuf に追加する．Double flag は，値が 0 であれば通常の演算モード用のデータが格納され，値が 1 の場合は 2 倍のビット幅を利用する 2 倍ビット長演算モード用のデータであることを示す．さらに，2 倍ビットデータを用いる際の下位ビットを格納する Half Weight Memory' を追加する．

Half Weight Memory からのデータをロードする際に Double flag をチェックし，その値が 0 の場合，図 4 に示すように各演算器に対応するラッチの上位半分のビット部に Half Weight Memory からのデータを格納する．また，下位ビット部には 0 を挿入する．一方で，Double flag の値が 1 の場合，すわなち 2 倍ビット長演算モード用のデータであった場合は，図 5 に示すように，ラッチの上位半分のビット部にいったんロードされたデータを格納し，下位ビット部に格納するべきデータを Half Weight Memory' からロードし，Half Weight Memory からのデータと連結して 1 データとする．その後演算が実行される．

例えば，各演算器の入力ビット幅が 16 ビットの場合は，Half Weight Memory 上の 1 エントリに保存されるデータ

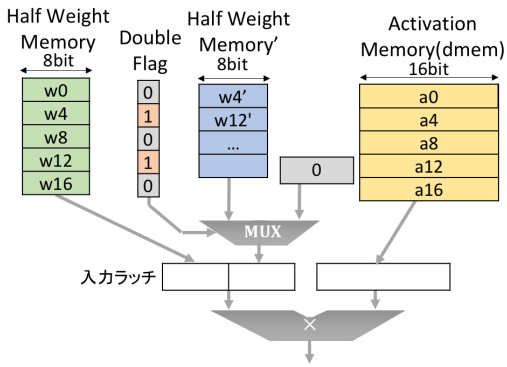

図 3 アーキテクチャの拡張

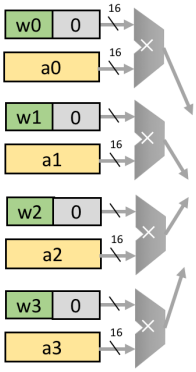

図 4 通常の演算 (xflag = 0)

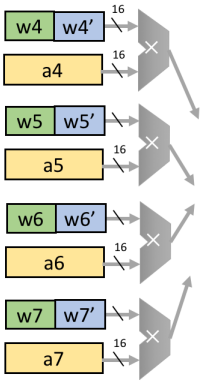

図 5 2 倍ビット長演算モード (xflag = 1)

長は 8 ビット ×SIMD 演算器数となり，各演算器には 8 ビットのデータが供給される．したがって，通常モードでは 8 ビットデータと下位 8 ビットに 0 が拡張された 16 ビットデータに対して演算が行われる．一方で，2 倍ビット長演算モードであれば，Half Weight Memory のデータと Half Weight Memory' のデータから 16 ビットデータを生成しそれに対して演算が行われる．

5. 評価実験

5.1 評価条件

ニューロン毎にビット幅を最適化することによるデータ量の削減効果と，画像認識精度への影響を確認するために評価実験を行う．

評価に用いたネットワークモデルは，ディープラーニングフレームワーク Chainer[14] の手書き文字認識サンプルのネットワークであり，3 層のパーセプトロンである．各層同士は全結合しており，ノード数は，入力層が 784，2 つの中間層である第 1 層，第 2 層がそれぞれ 1000，1000，出力層である第 3 層が 10 である．活性化関数には ReLU 関数を用いる．また，データセットには，MNIST[8] を用いた．最適化手法に Adam を用いて 30epoch の学習を行った結果，テスト精度は 98.36%であった．この精度を以下で述べる相対認識精度の基準値とする．

本稿では，実際に固定小数点で量子化や演算を行うのではなく，以下に示す方法で重みの量子化と固定小数点数の精度をエミュレートした．

まず，学習は 32 ビットの浮動小数点数で行われるため，32 ビットの浮動小数点数データに量子化したいビット数分に対応する 2 のべき乗数を掛けてから整数に変換し，再度浮動小数点数に変換する．実際の計算は 32 ビットの浮動小数点演算で行った．本評価での量子化の手順を，以下に示す．なお，小数点以下のビット幅を n ビットとする．

(1) 重みパラメータ (float 32-bit) に 2^n を乗算する
(2) 重みパラメータを int 16-bit に変換する
(3) 重みパラメータを float 32-bit に変換する
(4) 重みパラメータを 2^n で除算する

なお，学習済みの重みの値は全て 1 未満であり，小数点以下のみを保持すればよいため，小数部のみ量子化すると想定する．そのため，小数点以下のビット幅が n ビット，小数点以上は 0 ビット，符号ビットを 1 ビットとした．つまり，n_l や n_s は $n+1$ ビットとなる．したがって，本実験での最小のビット幅は 2 ビットであり，2 ビットのデータは $+0.5$ あるいは -0.5 および 0 を表現する．また，重みのビット幅を表すフラグビットはニューロン毎に保持するため，一般の DNN におけるフラグビットの個数は重みパラメータの数に比べて数千分の 1 程度と小さいため，無視してよいと仮定する．たとえば本実験で用いるネットワークの第 2 層は 1000×1000 の行列とみなすことができ，このうち必要なフラグビットは 1 行に対して 1 つであり，第 2 層の重みパラメータ数全体の 1/1000 程度しかない．

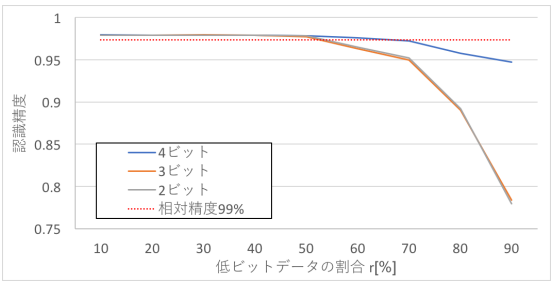

図 6 第 1 層のみニューロン毎の量子化を実施したときの認識精度（再学習なし）

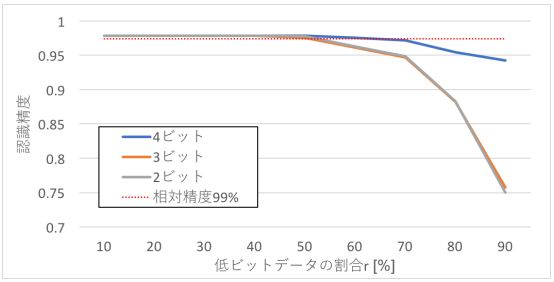

図 7 第 1 層にニューロン毎の，第 2 層，第 3 層にそれぞれ層毎の量子化を実施したときの認識精度（再学習なし）

5.2 第 1 層のみの評価実験

まず，第 1 層のみにニューロン毎の量子化を実施し，削減割合 r と短ビット幅 n_s を変化させ，ニューロン毎の量子化による認識精度の低下を評価する．このとき，第 2 層，

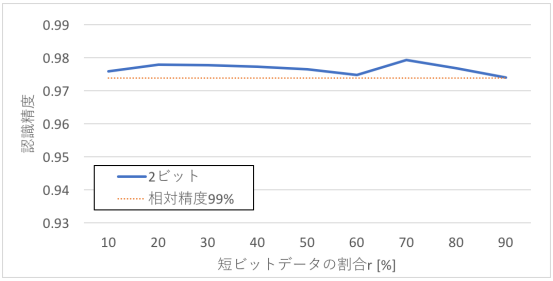

図 8 第 1 層のみニューロン毎の量子化を実施したときの認識精度（再学習あり）

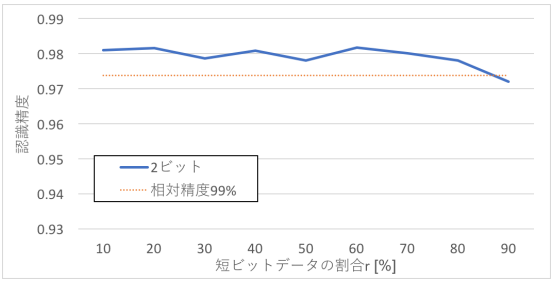

図 9 第 1 層にニューロン毎の，第 2 層，第 3 層にそれぞれ層毎の量子化を実施したときの認識精度（再学習あり）

第 3 層の重みは完全精度である．次に，第 2 層，第 3 層にそれぞれ層毎の量子化を実施した状態で第 1 層にニューロン毎の量子化を実施し，ネットワーク全体で精度が低下している状態からさらにデータ量を削減できるかを調査する．また，どちらの評価実験においても，再学習を実施しない場合と実施した場合の両方を実験し，再学習の効果を評価する．したがって，表 1 に示す 4 通りの実験を実施する．

表 1 実験の概要

実験	第 1 層	第 2 層	第 3 層	再学習
(1)	ニューロン毎量子化	完全精度	完全精度	無
(2)	ニューロン毎量子化	層毎量子化	層毎量子化	無
(3)	ニューロン毎量子化	完全精度	完全精度	有
(4)	ニューロン毎量子化	層毎量子化	層毎量子化	有

層ごとのビット幅最適化を行った結果を以下に示す．再学習なしの場合，$th = 0.99$ を満たす第 1 層，第 2 層，第 3 層の最小のビットは幅は，各 5 ビットであった．このときの認識精度は 97.49%，完全精度の重みを用いた場合に対する相対認識精度は 99.11%であった．再学習ありの場合は同様に，$th = 0.99$ を満たす第 1 層，第 2 層，第 3 層の最小ビット幅は各 3 ビットであり，このときの認識精度は 98.01%，完全精度時に対する相対認識精度は 99.65%であった．再学習を用いた場合はより大きくビット幅が削減できる余地があることが示された．

まず，実験 (1) の結果について述べる．第 2 層と第 3 層の重みのビット幅を完全精度，第 1 層の長ビット幅 n_l を 5 ビットとし，短ビット幅 n_s を 4 ビットから 2 ビットに，短ビット幅の割合 r を 10%から 90%にそれぞれ変化させたときの認識精度を図 6 に示す．$n_s = 4$ とした場合，相対認識精度が th を満たす最大の短ビットの割合 r は 60%であり，このときの認識精度は 97.80%，相対認識精度は 99.22%であった．同様に，$n_s = 3$ のときは $r = 50$ で認識精度が 97.73%，相対認識精度が 99.36%，また $n_s = 2$ のときは $r = 50$ で認識精度が 97.83%，相対認識精度が 99.46%であった．なお，相対認識精度が 99%以上であり最もデータ量が少ないのは，$n_s = 2$，$r = 50\%$ のときでああった．このときの第 1 層の重みの圧縮率は，完全精度の 10.94%であった．

次に，実験 (2) について述べる．第 2 層と第 3 層の重みのビット幅をそれぞれ 5 ビットとし，第 1 層の長ビット幅 n_l を 5 ビットとし，短ビット幅 n_s を 4 ビットから 2 ビットに，短ビット幅の割合 r を 10%から 90%にそれぞれ変化させたときの，認識精度を図 7 に示す．認識精度は，$n_s = 4$，$r = 60\%$ のとき 97.53%，$n_s = 3$，$r = 50\%$ のとき 97.56%，$n_s = 2$，$r = 50\%$ のとき 97.75%であった．

次に，実験 (3) と実験 (4) について述べる．第 2 層と第 3 層の重みのビット幅を完全精度とし，第 1 層の長ビット幅 n_l を 3 ビットとし，短ビット幅 n_s 2 ビットに，短ビット幅の割合 r を 10%から 90%にそれぞれ変化させ，再学習を実施したときの認識精度を図 8 に示す．また，第 2 層と第 3 層の重みのビット幅を完全精度とし，第 1 層の長ビット幅 n_l を 3 ビットとし，短ビット幅 n_s 2 ビットに，短ビット幅の割合 r を 10%から 90%にそれぞれ変化させ，再学習を実施したときの認識精度を図 9 に示す．図 8 を見ると，再学習を行った場合はいずれの割合でも相対認識精度が $th = 0.99$ を上回っており，大幅にビット幅を削減できることがわかる．一方図 8 では，$n_s = 2$，$r = 80\%$ までは相対認識精度が $th = 0.99$ を上回っており，以降の層を量子化しても，大幅にビット幅削減が可能である．

5.3 モデル全体での評価実験

前節では第 1 層のみでのニューロン毎のビット幅最適化を実行したが，第 2 層，第 3 層でも同様にニューロン毎のビット幅最適化を実施した．

まず，再学習なしの場合について述べる．再学習なしで層毎の量子化を行った結果のビット幅と完全精度に対する重みの圧縮率を表 2 に，再学習なしでニューロン毎の量子化行った結果を表 3 に示す．これらは，各量子化手法において，相対認識精度が 99%を保ちながら圧縮率が最も高かった結果である．このときの認識精度は，前者が 97.49%, 後者が 97.71%である．モデル全体の重みサイズを比較すると，ニューロン毎の量子化をした結果，層毎の量子化をした結果よりさらに重みのサイズを 40.1%削減することができ，より積極的な容量削減ができる可能性が示された．また，表 3 の各層でのビット幅削減量について注目すると，入力に近い方の層の方が大幅に重みを圧縮できていることがわかる．この原因として，第 1 層，第 2 層はそれぞれ重みパラメータの数が $784 \times 1000, 1000 \times 1000$ と第 3 層の 1000×10 個に比べて多く，過剰にパラメタライズされているため，不要な情報をより多く保持している可能性が考えられる．他には，量子化によって入力層の方から情報の欠落が大きくなり，出力層に近づくに従って累積し

た誤差が大きくなるため，大きく情報を削減することができない可能性も考えられる．これらを示すために，ネットワークのニューロン数や層の数などを変化させた場合に対する量子化の効果を検証する必要があると考えられる．

表 2 モデル全体を層毎の量子化した結果（再学習なし）

層	ビット幅	重みの圧縮率
第 1 層	5	15.652%
第 2 層	5	15.652%
第 3 層	5	15.652%
モデル全体		15.652%

表 3 モデル全体をニューロン毎の量子化した結果（再学習なし）

層	長ビット幅 n_l	短ビット幅 n_s	重みの圧縮率
第 1 層	5 (50%)	2(50%)	10.938%
第 2 層	5 (80%)	2(20%)	13.750%
第 3 層	5 (100%)	2(0%)	15.625%
モデル全体			12.531%

また，上記の評価に再学習を適用する実験を行った．結果をそれぞれ表 4，表 5 に示す．再学習をしながらモデル全体を層毎に量子化した結果，認識精度は 98.01%，モデル全体の重みの圧縮率は 9.375%であり，さらにニューロン毎の量子化を行った場合の認識精度は 97.75%，重みの圧縮率は 6.873%である．ニューロン毎の量子化は，層毎の量子化と比べて，重みのサイズを 27.7%削減した．また，このときの短ビット幅の割合をそれぞれ見ると，どの層においても 80%から 90%のニューロンの重みを削減できていることがわかる．このことから，低精度の重みの少量にデータを付加し，少ないデータ量増加で精度を向上させる手法などへの応用が考えられる．

表 4 モデル全体を層毎の量子化した結果（再学習あり）

層	ビット幅	重みの圧縮率
第 1 層	3	9.375%
第 2 層	3	9.375%
第 3 層	3	9.375%
モデル全体		9.375%

表 5 モデル全体をニューロン毎の量子化した結果（再学習なし）

層	長ビット幅 n_l	短ビット幅 n_s	重みの圧縮率
第 1 層	3 (20%)	2 (80%)	6.875%
第 2 層	3 (20%)	2 (80%)	6.875%
第 3 層	3 (10%)	2 (90%)	6.563%
モデル全体			6.873%

5.4 ビット幅を縮小するニューロン選択手法の評価

提案手法におけるビット幅を削減するニューロンの選択は，3.1 節で述べたように，完全精度と量子化後とで精度の差を評価する方法で行う．この精度の差が小さいニューロンから優先して短ビットになり，精度の差が大きいニューロンは長ビットにする．この手法は理論的な根拠に基づくものではないため，最も単純なニューロンの選択をランダムに行った場合との認識精度を比較してその有効性を検証する．

実験条件を示す．5.3 節において最も圧縮率の大きかった長ビット幅 $n_l = 4$, 短ビット幅 $n_s = 2$ の場合において，r の値を 0.1 から 0.9 まで 0.1 刻みで増加させたときの認識精度を，提案手法およびランダムでビット幅を削減するニューロンを選択した場合とで比較する．ニューロン毎の量子化を適用するのは第 1 層のみで，他の層の重みは完全精度で表現される．ランダムなニューロンの選択は，一様乱数によって生成した 100 個のインデックスに基いて行った．

実験の結果を表 6 に示す．表は，短ビット化する割合 r における，提案手法を用いた場合の認識精度と，ランダムにニューロンを選択した場合の平均認識精度，最高認識精度，最悪認識精度を示す．$r = 0.1$ から $r = 0.3$ の間では，提案手法よりもランダム選択の最高の場合の方が精度が高いが，その差はたとえば $r = 0.1$ のとき 0.28 ポイントなどと小さいことと，平均認識精度を上回っていることから，有効性が示唆される．一方で，r の値が 0.4 から 0.5, 0.6 と大きくなるに従って，提案手法がランダム選択の最高認識精度を上回るようになり，ランダムと比べて高い性能になっており，提案のニューロン選択手法の有効性が示された．

表 6 ビット幅削減するニューロンをランダムで選ぶ量子化と提案手法との比較（再学習なし）

r	提案手法	ランダム選択		
	認識精度	平均認識精度	最高認識精度	最悪認識精度
0.1	97.93%	97.75%	98.21%	97.30%
0.2	97.90%	97.45%	98.07%	96.41%
0.3	97.23%	96.91%	97.71%	94.34%
0.4	97.92%	96.03%	97.47%	92.09%
0.5	97.83%	94.47%	97.21%	87.81%
0.6	96.50%	91.53%	95.65%	83.79%
0.7	95.19%	85.58%	93.05%	72.00%
0.8	89.22%	71.49%	88.87%	53.66%
0.9	77.95%	41.08%	66.31%	19.11%

次に，再学習を行った場合について述べる．長ビット幅 $n_l = 2$, 短ビット幅 $n_s = 1$ の場合において，同様の実験を行った．ただし，再学習を行う計算時間の都合上，ランダム選択では前述の 100 個のインデックスから無作為に抽出した 10 個のインデックスをもとに，ニューロン毎の量子化を行った．実験結果を表 7 に示す．

再学習なしの場合とは対照に，再学習をした場合は提案手法の認識精度が平均を下回る場合が多くなっている．この理由として考えられるのが，評価対象のネットワークは全結合ネットワークであるため，再学習をする場合はいずれのニューロンのビット幅を削減しても，重みの初期値が異なる等価なネットワークであると解釈が可能であることである．また，その他に，このネットワークの第 1 層は 5.2 節で示した通り精度の低下を抑えながら 80%のニューロンのビット幅を削減することが可能であり，余分に削減できる情報が多くあるため，ニューロンの選択に差がないということが考えられる．

表 7 ビット幅削減するニューロンをランダムで選ぶ量子化と提案手法との比較（再学習あり）

r	提案手法	ランダム選択		
	認識精度	平均認識精度	最高認識精度	最悪認識精度
0.1	97.23%	97.38%	97.71%	96.76%
0.2	97.42%	97.29%	97.51%	96.89%
0.3	97.46%	97.39%	97.77%	97.24%
0.4	97.05%	97.28%	97.45%	96.97%
0.5	97.41%	97.30%	97.66%	97.00%
0.6	97.13%	97.29%	97.59%	96.80%
0.7	97.23%	97.16%	97.44%	96.72%
0.8	97,17%	97.06%	97.35%	96.71%
0.9	96.91%	96.70%	96.86%	96.34%

6. まとめ

本稿では，ニューロン毎の量子化を適用したネットワークに再学習を行った場合の評価や，量子化をモデル全体に適用する評価を行った．再学習を行うことで，再学習を行わないニューロン毎の量子化よりもモデル全体で 1.82 倍重みパラメータのサイズを削減できることが示された．また，モデル全体にニューロン毎の量子化を行う場合，層が出力層に近づくほどビット幅を削減できる割合が小さくなることが確認されたが，再学習を適用することで出力層に近い層でも大幅にビット幅が削減可能になった．今後の課題として，量子化の性質をより定量的に捉えるために，ネットワークのニューロン数や層の数などのハイパーパラメータを変化させたときの量子化の効果を評価することが挙げられる．また，ImageNet 等の，1 サンプルのサイズが大きく，サンプル数も多量である MNIST よりもより情報量の大きいデータセットでの評価を検討している．

謝辞 本研究の一部は JSPS 科研費基盤研究 (S)25220002，および JST CREST 課題番号 JPMJCR1785 の支援を受けたものである．

参考文献

[1] Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., and Temam, O.: *Dadiannao: A machine-learning supercomputer*, Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 609-622) (2014).

[2] Chen, Y. H., Krishna, T., Emer, J. S., and Sze, V.: *Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks*, IEEE Journal of Solid-State Circuits, 52(1), 127-138 (2017).

[3] Han, S., Pool, J., Tran, J., and Dally, W.: *Learning both weights and connections for efficient neural network*, Advances in Neural Information Processing Systems (pp. 1135-1143) (2015).

[4] Han, S., Mao, H., and Dally, W. J.: *Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding*, arXiv preprint arXiv:1510.00149 (2015).

[5] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. *EIE: efficient inference engine on compressed deep neural network* Proceedings of the 43rd International Symposium on Computer Architecture (pp. 243-254) (2016).

[6] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y.: *Binarized neural networks*, Neural Information Processing Systems (pp. 4107-4115) (2016).

[7] Krizhevsky, A., Sutskever, I., and Hinton, G. E.: *Imagenet classification with deep convolutional neural networks*, In Advances in neural information processing systems (pp. 1097-1105) (2012).

[8] LeCun, Y.: *The MNIST database of handwritten digits*, http://yann. lecun. com/exdb/mnist/.

[9] Lee, E., Miyashita, D., Chai, E., Murmann, B. and Wong, S. *LogNet: Energy-efficient Neural Networks using Logarithmic Computation*, International Conference on Acoustics, Speech and Signal Processing (2017).

[10] Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S.: *Recurrent neural network based language model*, In Interspeech (Vol. 2, p. 3) (2010).

[11] Miyashita, D., Lee, E. H., and Murmann, B.: *Convolutional neural networks using logarithmic data representation*, arXiv preprint arXiv:1603.01025 (2016).

[12] Moons, B., De Brabandere, B., Van Gool, L., and Verhelst, M. (2016, March): *Energy-efficient ConvNets through approximate computing*, Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on (pp. 1-8) (2016).

[13] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A.: *Xnor-net: Imagenet classification using binary convolutional neural networks*, European Conference on Computer Vision (pp. 525-542). Springer International Publishing (2016).

[14] Tokui, S., Oono, K., Hido, S., Clayton, J.: *Chainer: a Next-Generation Open Source Framework for Deep Learning*, In Workshop on Machine Learning Systems at Neural Information Processing Systems (2015).

[15] Vanhoucke, V., Senior, A., and Mao, M. Z.: *Improving the speed of neural networks on CPUs*, Deep Learning and Unsupervised Feature Learning NIPS Workshop (Vol. 1, p. 4) (2011).

[16] Zhu, C., Han, S., Mao, H., and Dally, W. J.: *Trained Ternary Quantization*, arXiv preprint arXiv:1612.01064 (2016).

[17] Courbariaux, M., Bengio, Y., and David, J.-P. BinaryConnect: *Training Deep Neural Networks with binary weights during propagations*, Advances in Neural Information Processing Systems, pp. 3105-3113, (2015).

[18] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi: *HARDWARE-ORIENTED APPROXIMATION OF CONVOLUTIONAL NEURAL NETWORKS*, arXiv preprint arXiv:1604.03168 (2016).

[19] 高田, 石井, 坂本, 近藤, 中村, 大久保, 小島, 天野, : スケーラブルなディープラーニング向けアクセラレータチップの設計と評価, 情報処理学会研究報告, Vol. 2016-ARC-223, No. 1, (2016).

[20] 石井, 坂本, 近藤, “可変データビット幅を持つ DNN とそのアクセラレータアーキテクチャの検討”, 情報処理学会研究報告 Vol.2017-ARC-227, No.25, (2017).