社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Binarized-DCNN による識別計算の高速化とモデル圧縮

神谷　龍司†　山下　隆義†　安倍　満††　佐藤　育郎††　山内　悠嗣†
藤吉　弘亘†

† 中部大学 〒 487–8501 愛知県春日井市松本町 1200
†† 株式会社デンソーアイティーラボラトリ 〒 150-0002 東京都渋谷区渋谷二丁目 15 番 1 号 渋谷クロスタワー 28 階
E-mail: †shinryu@vision.cs.chubu.ac.jp, ††{yamashita,hf}@cs.chubu.ac.jp, †††{manbai,isato}@d-itlab.co.jp, ††††yuu@isc.chubu.ac.jp

あらまし　Deep Convolutional Neural Network (DCNN) は，一般物体認識やセマンティックセグメンテーション等の様々なタスクにおいて高い認識性能を実現した．DCNN は計算量とパラメータ数が非常に多いため，膨大な識別時間と多大なメモリ量を必要とする．層をより深くすることで認識精度が向上する傾向が見られ，これに伴い識別時間に加えモデルサイズも増加するという問題が発生する．組み込みやモバイル機器等の低スペックのデバイスで使用するには，識別計算の高速化とモデルサイズの圧縮が大きな課題となる．本研究では，これらの問題を再学習なしで解決する Binarized-DCNN を提案する．提案手法は，Quantization sub-layer と結合係数の二値分解を導入し，実数同士の内積計算を二値同士の内積計算に置き換えることで，既存のネットワークモデルに対して再学習なしに識別計算の高速化及びモデルサイズの圧縮を実現する．二値同士の演算は XOR や AND などの論理演算とビットカウントにより高速な演算が可能となる．評価実験により，エラー増加率を 2.0%程度に抑え，約 2.0 倍の高速化と約 80%のモデル圧縮を達成した．

キーワード　Deep Convolutional Neural Network，識別計算の高速化，モデルサイズの圧縮

1. はじめに

Deep Convolutional Neural Network (DCNN) は，一般物体認識やセマンティックセグメンテーションを始めとした様々なタスクにおいて高い認識性能を実現した．1000 カテゴリ識別を対象とする ImageNet Large Scale Visual Recognition Challenge (ILSVRC) では，2012 年に 8 層の DCNN である AlexNet [9] が優勝して以来，16 層の VGG-16 [12] や 152 層の Residual Network (ResNet) [6] といった膨大な層数を持つネットワークモデルが登場し，驚異的な速度でエラー率を低減し続けている．識別精度の向上や難しいタスクを識別するには，ネットワークの多層化あるいは複雑化するという傾向がある．モデルの複雑化に伴い，識別時間やモデルサイズの増加といった問題が発生する．組み込みやモバイル機器など低いリソース環境で利用する際には，識別計算の高速化やモデルサイズの圧縮は必要不可欠である．この問題を解決するために，識別計算の高速化 [8] [14] やモデルサイズの圧縮 [5] に関する研究が提案されている．

ただし，これらの手法は高速かつ必要なメモリサイズを効果的に削減できるが，モデルを再度学習する必要があるため，既存ネットワークモデルに対して再学習なしに適用することができない．

そこで本研究では，既存ネットワークモデルに対して再学習なしに識別計算の高速化とモデルサイズの圧縮が可能な Binarized-DCNN を提案する．Binarized-DCNN は，識別計算に用いる特徴マップと各層の重みを二値に変換し，近似内積計算を行うことで既存のネットワークモデルにおける識別計算の高速化とモデルの圧縮を同時に実現する．提案手法の特長を以下に示す．

- 実数値のパラメータを多数の二値と少数の実数値で近似することで再学習なしに識別計算の高速化とモデル圧縮を同時に達成
- Quantization sub-layer を導入することで実数の特徴マップを高速に二値の特徴マップに変換
- BinaryNet や XNOR-Net では精度低下が大きい大規模なネットワークモデルに対して適用可能

2. 関連研究

本章では，DCNN における識別計算の高速化やモデルサイズの圧縮に関する従来研究について述べる．

2.1 低ランク近似による識別計算の高速化

低ランク近似による識別計算の高速化 [8] [14] は，近似誤差が最小になるように各層の重みを近似することでエラー率の増加を抑え高速化する．[8] は，重みを 2 種類の 1 次元重みフィルタ

に分解することで順伝播の計算を高速化する．[14] は，応答値の低ランク近似を浅層から順番に適用することで計算量を削減し高速化する．一般に，浅層で発生した近似誤差は識別精度に大きく影響する．[14] では重みを近似した際の応答値を次層で近似することで近似誤差の影響を低減する．

2.2 パラメータ削減によるモデル圧縮

モデル圧縮の研究には Deep Compression [5] がある．Deep Compression は枝刈りと量子化，ハフマン符号化の 3 つを組み合わせることでモデルサイズを約 1/50 に圧縮しつつ性能を向上している．まず，学習済みモデルにおいて，あるしきい値以下の重みを全て 0 にして結合を削除する．次に，疎行列とした重みにクラスタリングを施して類似した重みを共有化する．最後に，ハフマン符号化によりモデルサイズを圧縮する．共有重みのインデックスの分布は偏っているため，より効率良くメモリ消費を抑えることができる．

2.3 パラメータの二値化による識別計算の高速化とモデル圧縮

DCNN の特徴マップや重みを二値化することで識別計算の高速化とモデル圧縮を同時に達成する手法として BinaryNet [3] [7] と XNOR-Net [10] がある．BinaryNet は，パラメータを BinaryConnect [4] により二値表現することでメモリサイズの圧縮と高速に内積計算を行うことができる．活性値と重みを二値にすることで識別計算の高速化とモデルの圧縮が同時に達成できるが，誤差逆伝播法によるパラメータの更新量が算出できない．そこで BinaryNet では Straight-through estimator [2] により勾配の計算を可能にする．XNOR-Net は，重みの近似にスケーリング係数を導入することで BinaryNet よりも高精度化している．そのため，XNOR-Net では各重みの近似誤差が最小になるようにバイナリフィルタとスケール係数の 2 つを最適化する．パラメータの更新量の算出には，BinaryNet と同様の方法を利用する．また，畳み込み計算時において内積計算が重複している領域が存在する．これはチャンネル方向に絶対値の平均を計算し，その出力とバイナリフィルタの畳み込み演算を行うことで効率化する．

これまでに述べた手法は，学習あるいは学習済みのパラメータを再度更新する必要がある．しかし，DCNN のパラメータの更新は多大な時間を必要とする．そこで学習済みの実数パラメータをベクトル分解により二値パラメータに変換することを考える．

3. ベクトル分解

3.1 実数ベクトルから二値基底行列への分解

二値同士の内積計算を行うためには，実数パラメータを二値に変換する必要がある．パラメータを二値に変換する方法にベクトル分解 [11] [13] [1] がある．ベクトル分解は，重みベクトル $\mathbf{w} \in \mathbb{R}^D$ を二値基底行列 $\mathbf{M} \in \{-1,1\}^{D\times k}$ とスケール係数ベクトル $\mathbf{c} \in \mathbb{R}^k$ に分解する．ここで，k は基底数，D は入力次元数を示している．重みベクトルに対してベクトル分解を適用することで，入力ベクトル $\mathbf{x}$ が二値の場合において実数同士の内積計算を二値同士の内積計算に置き換えることができる．二値同士の内積計算は論理演算とビットカウントにより高速に演算が可能である．ベクトル分解の最適化手法に greedy アルゴリズムによる最適化 [11] と exhaustive アルゴリズムによる最適化 [13] がある．本章では，近似精度の良い exhausetive アルゴリズムによる最適化を用いてベクトル分解を行う．

3.2 Exhaustive アルゴリズム

Exhaustive アルゴリズムによる分解は，重みベクトル $\mathbf{w}$ を式 (1) のコスト関数を最小化する二値基底行列 $\mathbf{M}$ とスケール係数ベクトル $\mathbf{c}$ に分解する．$\mathbf{M}$ を全探索により最適化するため分解に膨大な時間を要するが，greedy アルゴリズムよりも近似精度の良い分解が可能である．分解アルゴリズムを Algorithm1 に示す．まず，$\mathbf{M}$ を $\{-1,1\}$，$\mathbf{c}$ を実数の乱数により初期化する．次に，$\mathbf{M}$ と $\mathbf{c}$ を最適化する．このとき，$\mathbf{M}$ と $\mathbf{c}$ を同時に最適化することは困難である．そのため $\mathbf{M}$ と $\mathbf{c}$ を交互に最適化することを考える．$\mathbf{M}$ を固定し，式 (1) を最小化する $\mathbf{c}$ を最小二乗法により求める．次に，$\mathbf{c}$ を固定し，式 (1) を最小化する $\mathbf{M}$ を全探索により最適化する．これらの処理を式 (1) のコスト関数の値が収束するまで繰り返す．なお，ベクトル分解の近似精度は初期値に依存するため，初期値を L 回変更して式 (1) が最も小さい $\mathbf{M}$ と $\mathbf{c}$ をその基底の分解結果とする．

$$E = ||\mathbf{w} - \mathbf{M}\mathbf{c}||_2^2 \tag{1}$$

Algorithm 1 分解アルゴリズム

Require: $\mathbf{w}$, k, L
  for i to L **do**
    $\mathbf{M}_i$ を $\{-1,1\}$ の乱数により初期化
    $\mathbf{c}_i$ を実数の乱数により初期化
    repeat
      $\mathbf{c}_i = (\mathbf{M}_i^{\mathrm{T}}\mathbf{M}_i)^{-1}(\mathbf{M}_i^{\mathrm{T}}\mathbf{w})$
      $\mathbf{M}_i = \underset{\mathbf{M}_i\in\{-1,1\}^{D\times k}}{\arg\min} ||\mathbf{w} - \mathbf{M}_i\mathbf{c}_i||_2^2$
    until 式 (1) が収束
  end for
  $\hat{\mathbf{M}}, \hat{\mathbf{c}} = \underset{\mathbf{M},\mathbf{c}}{\arg\min} ||\mathbf{w} - \mathbf{M}\mathbf{c}||_2^2$
  return $\hat{\mathbf{M}}, \hat{\mathbf{c}}$

4. 提案手法

提案手法は，DCNN における識別計算の高速化とモデル圧縮を同時に達成するために，各層の特徴マップと重みを二値に変換し論理演算とビットカウントによる近似内積計算を行う．このとき，各層の重みの大半を二値に変換するため，モデルサイズの圧縮を同時に行うことができる．ただし，二値に変換した重みを用いて近似内積計算を行うには，入力特徴マップも二値である必要がある．しかし，特徴マップは入力サンプル毎に異なる値を持つため，ベクトル分解のように事前に変換することは困難である．提案手法では，特徴マップを高速に二値に変換することができる Quantization sub-layer を導入することで，高速に実数の特徴マップを二値の特徴マップに変換する．

4.1 ベクトル分解による重みの分解

提案手法では，近似精度の高い exhaustive アルゴリズムによ

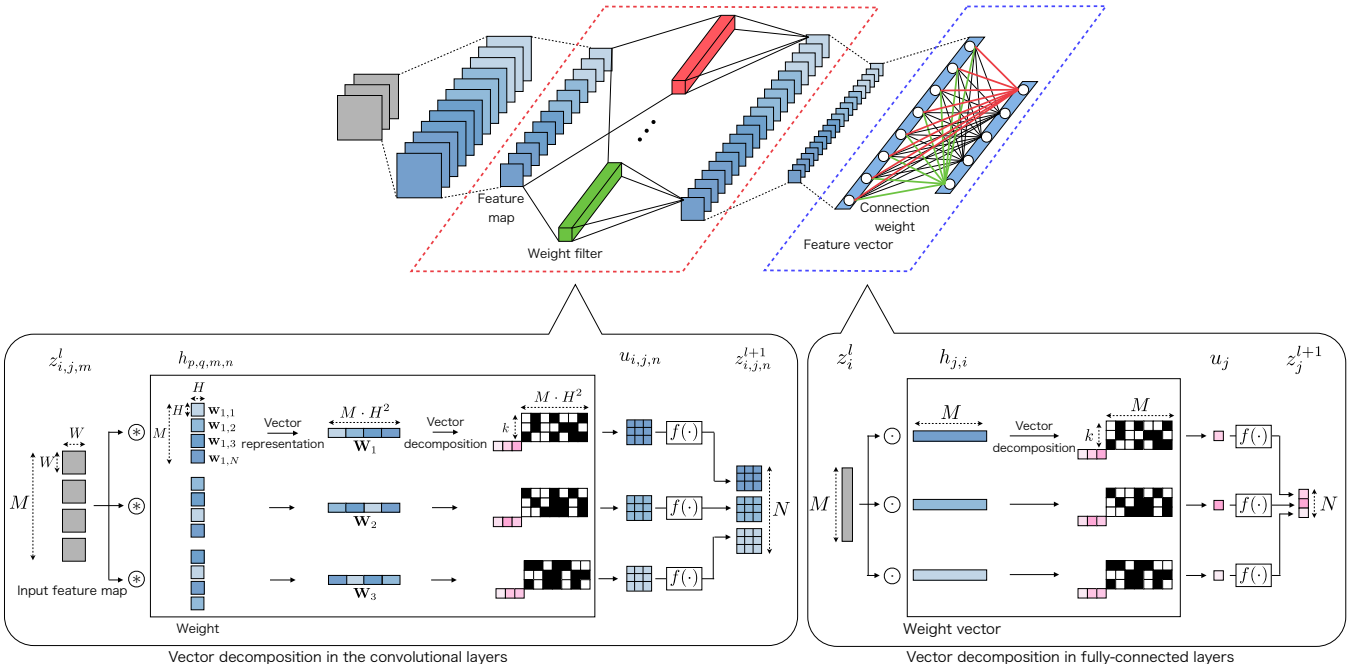


図 1　各層における重みの分解方法

るベクトル分解を用いて畳み込み層と全結合層の重みを二値に変換する．二値に変換することで近似内積計算による識別計算の高速化と，各パラメータを 1 ビットの集合で表現することができるためモデルの圧縮を同時に行うことができる．各層における重みの分解方法を図 1 に示す．

4.1.1　畳み込み層への適用

まず，第 l 層における n 番目の特徴マップに繋がる m 番目の重みフィルタ $\mathbf{w}^l_{n,m} \in \mathbb{R}^{H\times H}$ に対してベクトル分解を適用することを考える．ベクトル分解はベクトルを対象とした分解手法であるため，行列である重みフィルタに直接適用できない．そこで提案手法では，重みフィルタをベクトル表現することで分解法の適用を可能にする．重みフィルタをベクトル表現することにより H^2 次元のベクトルとなる．しかし，VGG-16 や ResNet のようなネットワークモデルは，1 枚当たりの重みフィルタサイズが非常に小さい場合が多い．この場合，計算量を削減できないため近似内積計算による効果が得られない場合がある．そこで，分解する重みフィルタを 1 枚だけではなくチャンネル方向のフィルタも利用してベクトル表現する．入力特徴マップ数を M とすると，分解する重みベクトルは $\mathbf{W}_n \leftarrow \{\mathbf{w}_{n,1}, \mathbf{w}_{n,2}, ..., \mathbf{w}_{n,M}\} \in \mathbb{R}^{M\cdot H^2}$ と定義できる．このとき，$\mathbf{W}_n$ の次元数は $M\cdot H^2$ となるため，近似内積計算の効果が得られやすくなる．また，畳み込み層には $\mathbf{W}$ が出力マップ数 N 個だけ存在する．各 $\mathbf{W}$ に対して Algorithm1 によるベクトル分解を適用し，$\hat{\mathbf{M}}$ と $\hat{\mathbf{c}}$ に分解する．畳み込み層の Inference 処理では，この $\hat{\mathbf{M}}$ と $\hat{\mathbf{c}}$ を用いて近似内積計算を行う．

4.1.2　全結合層への適用

次に，全結合層の第 l 層における n 番目のユニットに繋がる結合重み $\mathbf{w}^l_n \in \mathbb{R}^M$ に対してベクトル分解を適用することを考える．全結合層において，n 番目のユニットにおける出力を得るために用いられる重みは M 次元の重みベクトルである．また，全結合層において $\mathbf{w}^l$ は出力ユニット数 N だけ存在する．各重みベクトル $\mathbf{w}^l$ に対して Algorithm1 によるベクトル分解を適用し，二値基底行列 $\hat{\mathbf{M}}$ とスケール係数ベクトル $\hat{\mathbf{c}}$ に分解する．

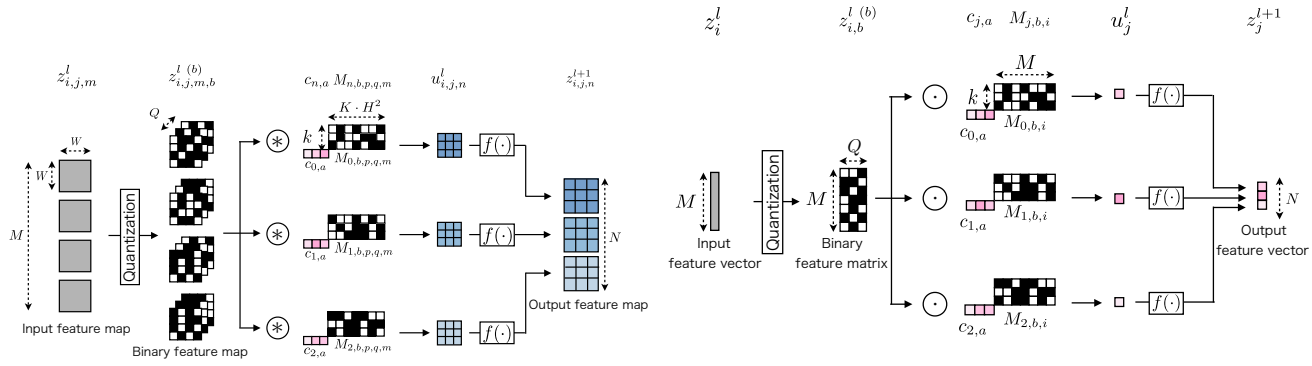
4.2　Quantization sub-layer

DCNN の特徴マップは負値を含んだ実数値であるため，近似内積計算を行うためにこれを二値の特徴マップに変換する必要がある．ただし，各層における特徴マップの値は入力サンプルに依存するため，ベクトル分解のように事前に量子化することは困難である．そのため識別計算時に特徴マップを変換しなければならず，変換に要する時間が DCNN の識別時間に直接影響する．このことから，実数の特徴マップを高速に二値の特徴マップに変換する必要がある．そこで，実数値を高速に二値へ変換する Quantization sub-layer を導入する．Quantization sub-layer は，量子化幅を可変にすることで負値を含んだ実数値の量子化を可能にする．

特徴マップを量子化する前に，事前準備として量子化ビット数 Q を設定する．Q が大きいほど量子化の近似精度は向上するが，近似内積計算に必要とする計算量が増加するため識別計算は遅くなる．逆に Q が小さいほど計算量が減少するため識別計算は高速になるが，量子化による近似精度は低下する．まず，式 (2) より特徴マップ $z_{i,j,m}$ における最大-最小値間の量子化幅 Δd を求める．このとき，Δd は特徴マップの最大値と最小値に依存するため，特徴マップ毎に異なる値を持つ．

$$\Delta d = \frac{\max(z^l_{i,j,m}) - \min(z^l_{i,j,m})}{2^Q - 1} \tag{2}$$

次に，式 (3) を用いて特徴マップの最小値が 0 になるように値をシフトさせる．特徴マップをシフトさせることで，本来量子化できない負値であっても量子化が可能となる．



(a) 畳み込み層　　(b) 全結合層

図 2　各層における Inference 処理

$$z^{l}_{i,j,m}{}' = \frac{z^{l}_{i,j,m} - \min(z^{l}_{i,j,m})}{Q} \tag{3}$$

最後に，$z^{l}_{i,j,m}{}'$ を量子化する．実数値は量子化できないため，四捨五入を施して整数値に丸め量子化する．$z^{l}_{i,j,m}{}'$ を量子化をすることでバイナリコード $z^{l\ (b)}_{i,j,m,b} \in \{0,1\}$ が生成される．

4.3　Inference 処理

Binarized-DCNN は二値同士の近似内積計算により識別計算を高速化する．二値同士の演算を行うために，Quantization sub-layer を用いて特徴マップ $z^{l}_{i,j,m}$ を二値化することを考える．しかし各層における特徴マップ及び特徴ベクトルの値は入力サンプルに依存するため，重みをベクトル分解する場合とは異なり，事前に量子化できない．Quantization sub-layer は実数値を高速に二値に変換できるという特徴を有しているため，識別計算時に量子化することができる．

4.3.1　畳み込み層における識別計算

通常の畳み込み計算は，実数値である特徴マップ $z_{i,j,m}$ の局所領域と重みフィルタ $h_{p,q,m,n}$ の積和を計算することで n 番目の特徴マップ $u_{i,j,n}$ の出力が得られる．本手法では，実数値による積和計算を二値に置き換えて出力を計算する．畳み込み層における識別計算を図 2(a) に示す．まず，入力特徴マップを Quantization sub-layer により量子化する．これにより，二値特徴マップ $z^{l\ (b)}_{b,i} \in \{0,1\}^{Q\times M\times W\times W}$ が生成される．その後，二値特徴マップとベクトル分解を施して得られた二値重みフィルタ $M_{n,b,p,q,m}$ と $c_{m,a}$ を用いて畳み込み計算を行う．l 層目の出力 $z^{l+1}_{i,j,n}$ は式 (5) となる．ここで，$\hat{M}_{n,b,p,q,m}$ はベクトル分解により得られた二値基底行列であり，$\hat{c}_{n,a}$ はスケール係数ベクトル，Δd は式 (2) より得られた量子化幅である．

$$z^{l+1}_{i,j,n} = f\left(u^{l}_{i,j,n}\right) \tag{4}$$

$$\begin{aligned} u^{l}_{i,j,n} &= \sum_{m=0}^{M-1}\sum_{p=0}^{H-1}\sum_{q=0}^{H-1} z^{l}_{i+p,j+q,m}\cdot h_{p,q,m,n} \\ &\approx \sum_{a=0}^{k-1}\hat{c}_{n,a}\ \Delta d \sum_{b=0}^{Q-1} 2^{b} \sum_{m=0}^{M-1}\sum_{p=0}^{H-1}\sum_{q=0}^{H-1} \\ &\quad \hat{M}_{n,b,p,q,m}\cdot z^{l\ (b)}_{i+p,j+q,m,b} + \min(z^{l}) \end{aligned} \tag{5}$$

4.3.2　全結合層における識別計算

全結合層では実数値の特徴ベクトル z^{l}_{i} と結合重み $h_{j,i}$ の内積計算により u_j の出力が得られる．全結合層における内積計算を高速化するために，畳み込み層と同様に実数値を二値に置き換えて内積計算を行う．全結合層における識別計算を図 2(b) に示す．まず，入力特徴ベクトルを Quantization sub-layer により量子化する．これにより，入力特徴ベクトル $z^{l} \in \mathbb{R}^{M}$ から二値特徴行列 $z^{l\ (b)} \in \{0,1\}^{M\times Q}$ が生成される．そして，出力ユニット j の出力を二値特徴行列と事前に分解した二値の重みベクトルを用いて式 (6) で近似計算する．

$$\begin{aligned} u^{l}_{j} &= \sum_{i=0}^{M-1} h_{j,i} z^{l}_{i} \\ &\approx \sum_{a=0}^{k-1}\hat{c}_{j,a}\ \Delta d \sum_{b=0}^{Q-1} 2^{b} \sum_{i=0}^{M-1} \hat{M}_{j,b,i} z^{l\ (b)}_{b,i} + \min(z^{l}) \end{aligned} \tag{6}$$

このとき，$\hat{M}_{j,b,i} \in \{-1,1\}$ と $z^{l\ (b)}_{b,i} \in \{0,1\}$ は二値であるため，式 (7) を用いて論理演算とビットカウントで計算することができる．このとき，ビットカウントは Streaming SIMD Extensions (SSE) 4.2 に実装されている POPCNT 関数を使用することで高速に演算が可能である．

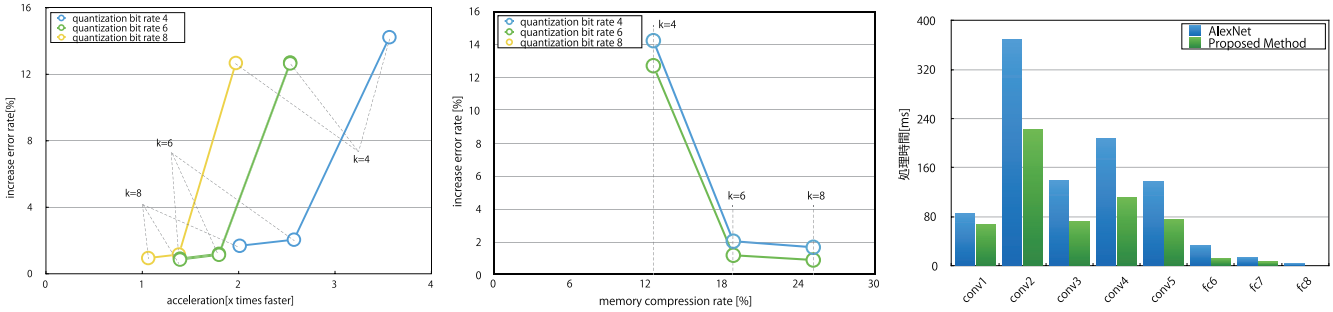
$$\begin{aligned} \hat{M}^{l}_{j,b,i} z^{l\ (b)}_{b,i} = 2\times \mathrm{POPCNT}(\mathrm{AND}(\hat{M}^{l}_{j,b,i}, z^{l\ (b)}_{b,i})) \\ -||z^{l}_{i}|| \end{aligned} \tag{7}$$

5.　評 価 実 験

提案手法を複数のネットワークモデルに適用した際の認識精度と処理時間及びモデルサイズを評価する．

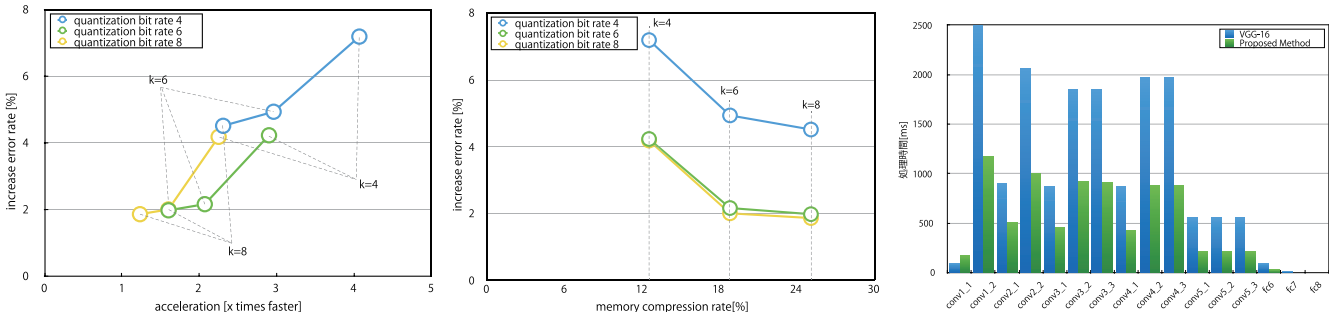
5.1　ImageNet classification task

ILSVRC2012 の分類タスクで使用された ImageNet を用いた評価を行う．ImageNet は学習サンプルが 1,200,000 枚，評価サンプルが 100,000 枚，検証サンプルが 50,000 枚存在する大規模な物体認識データセットで，各画像は 1,000 カテゴリのいずれかに分類される．本実験では，検証サンプル 50,000 枚を識別した際の Top5 Accuracy を用いて認識性能を評価する．Top-5 Accuracy は教師信号と同じ推定クラスの確率が上位 5 位以内であれば認識成功とする方法である．ネットワークモデルには AlexNet と VGG-16，及び ResNet-152 を用いる．各モデルのパラメータには公開されている学習済みモデルを使用



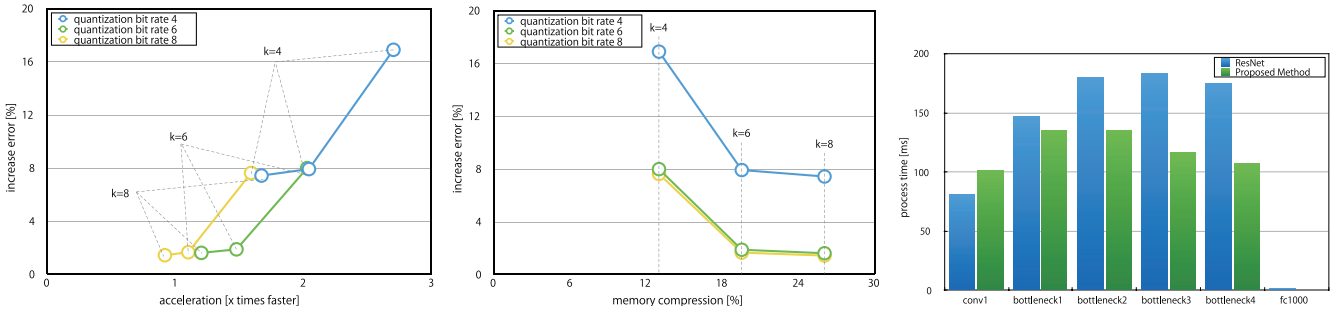
(a) error vs acceleration　(b) error vs memory compression　(c) process time

図 3　AlexNet の評価結果



(a) error vs acceleration　(b) error vs memory compression　(c) process time

図 4　VGG-16 の評価結果



(a) error vs acceleration　(b) error vs memory compression　(c) process time

図 5　ResNet-152 の評価結果

し，Fine tuning はしないものとする．近似計算に用いる量子化ビット数 Q は 4, 6, 8，基底数 k も同様に 4, 6, 8 を用いる．使用するプロセッサは Intel Core i7–4770 3.40GHz である．

5.1.1　モデル 1: AlexNet

AlexNet は 5 層の畳み込み層と 3 層の全結合層により構成される．本実験では，全層にわたって近似計算を導入するものとする．AlexNet に近似計算を導入した際の評価結果を図 3 に示す．ここで，k は重みの分解に用いた基底数を表している．図 3(a) より，基底数あるいは量子化ビット数を大きくすることで認識精度が向上する傾向が見られる．ただし，同基底数の量子化ビット数 6 と 8 を比較すると，ほとんどエラー増加率が変化していない．このことから，より計算量の少ない量子化ビット数 6 が最適であると言える．図 3(b) より，同基底数の量子化ビット数 4 と 6 においてメモリ圧縮率が変化していないことから，量子化ビット数はメモリ圧縮率に影響しない．図 3(c) より，近似計算を導入した全層において識別時間を短縮できていることがわかる．量子化ビット数 6，基底数 6 のとき，約 1.79 倍の高速化とモデルサイズを約 237.91MB から約 44.85MB までに (約 80%) 圧縮できた．このとき，エラー増加率は約 1.20%である．

5.1.2　モデル 2: VGG-16

VGG-16 は 13 層の畳み込み層と 3 層の全結合層により構成される．VGG-16 における畳み込み層 1 層目の次元数は非常に小さいため，近似内積計算の効果が得られない恐れがある．そのため本実験では，1 層目以外の全層に対して近似計算を導入するものとする．VGG-16 の全層に対して近似計算を施した際の識別時間を図 4(c) に示す．図 4(c) より，近似計算を導入した畳み込み層 1 層目以外の層においては識別時間を短縮できていることがわかる．近似計算適用前と適用後における畳み込み層 1 層目の処理時間を比較すると，適用前のほうが処理時間が短い．畳み込み層 1 層目のパラメータ数は非常に少ないため，近似内積計算を行うことで計算量が増加したためと考えられる．量子化ビット数 6，基底数 6 のとき，約 2.07 倍の高速化とモデルサイズを 527.74MB から 99.26MB までに (約 81%) 圧縮で

きた．このときのエラー増加率は 2.16%である．

5.1.3 モデル 3: ResNet-152

ResNet-152 は 151 層の畳み込み層と 1 層の全結合層により構成される．本実験では，全層にわたって近似計算を導入するものとする．近似内積計算における認識精度と処理時間，モデルサイズの比較を図 5 に示す．図 5(a) より，152 層のネットワークモデルである ResNet に対しても高速化できていることがわかる．続いて，エラー増加率とメモリ圧縮の関係を図 5(b) に示す．AlexNet や VGG-16 は全結合層が 3 層存在するため，全結合層がモデルサイズの 90%以上を占めていた．ResNet では畳み込み層が 152 層と非常に多いため，畳み込み層がモデルサイズの約 96%を占めている．ResNet のように畳み込み層が非常に多いモデルに対してもモデル圧縮の効果が得られることがわかる．量子化ビット数 6，基底数 6 のとき，約 1.77 倍の高速化とモデルサイズを約 229.08MB から約 44.71MB に (約 81%) 圧縮できた．このとき，エラー増加率は約 1.86%である．

5.2 Places scene recognition task

シーン認識のデータセットである Places [15] を用いた評価を行う．Places は学習サンプルが 2,448,873 枚，評価サンプルが 41,000 枚，検証サンプルが 20,500 枚からなるデータセットで，各画像は 205 シーンのラベルが付与されている．本実験では，検証サンプル 20,500 枚を用いて各ネットワークの性能を評価する．ネットワークモデルには AlexNet と VGG-16 を用いる．

評価結果を表 1 に示す．AlexNet では，量子化ビット数 6，基底数 6 のとき, 約 1.56 倍の高速化とモデルサイズを約 225.50MB から約 42.51MB に (約 81%) 圧縮できた．このとき，エラー増加率は約 3.44%である．VGG-16 では，量子化ビット数 6，基底数 6 のとき，約 1.69 倍の高速化と 515.30MB から 96.9MB に (約 81%) 圧縮できた．このとき，エラー増加率は 2.72%である．

表 1　Places を用いた評価結果

model	acceleration [x times faster]	memory compression rate [%]	increase error rate [%]
AlexNet	1.56	81	2.72
VGG-16	1.69	81	3.44

6. おわりに

本稿では，既存のネットワークモデルに対して再学習なしに識別計算の高速化とモデルを圧縮する方法を提案した．本手法は，各層の重みを実数のパラメータから二値のパラメータに変換することでメモリの圧縮を行い，実数同士の内積計算を論理演算とビットカウントを用いた二値同士の内積計算に置き換えることで識別計算の高速化を行っている．量子化ビット数 6，基底数 6 のとき，AlexNet ではモデルサイズを約 80%圧縮し，約 1.79 倍の高速化を実現した．このときのエラー増加率は約 1.20%である．VGG-16 では，モデルサイズを約 81%圧縮し，約 2.07 倍の高速化を達成した．このときのエラー増加率は約 2.16%である．今後の予定として，近似精度の向上によるエラー増加率の低減を目標とする．

文　献

[1] Mitsuru Ambai and Ikuro Sato. SPADE: Scalar Product Accelerator by Integer Decomposition for Object Detection. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Vol. 8693 of *Lecture Notes in Computer Science*, pp. 267–281. Springer, 2014.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, Vol. abs/1308.3432, , 2013.

[3] Matthieu Courbariaux and Yoshua Bengio. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *CoRR*, Vol. abs/1602.02830, , 2016.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. *CoRR*, Vol. abs/1511.00363, , 2015.

[5] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *CoRR*, Vol. abs/1510.00149, , 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, Vol. abs/1512.03385, , 2015.

[7] Itay Hubara, Daniel Soudry, and Ran El Yaniv. Binarized Neural Networks. *CoRR*, Vol. abs/1602.02505, , 2016.

[8] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

[10] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *CoRR*, Vol. abs/1603.05279, , 2016.

[11] Philip H.S. Torr Sam Hare, Amir Saffari. Efficient Online Structured Output Learning for Keypoint-Based Object Tracking. *the Proceedings IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, Vol. abs/1409.1556, , 2014.

[13] Yuji Yamauchi, Ambai Mitsuru, Sato Ikuro, Yuichi Yoshida, and Hironobu Fujiyoshi. Distance Computation Between Binary Code and Real Vector for Efficient Keypoint Matching. *Information Processing Society of Japan Transactions on Computer Vision and Applications*, Vol. 5, pp. 124–128, 2013.

[14] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating Very Deep Convolutional Networks for Classification and Detection. *CoRR*, Vol. abs/1505.06798, , 2015.

[15] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning Deep Features for Scene Recognition using Places Database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pp. 487–495. Curran Associates, Inc., 2014.