Vol.2018-ARC-232 No.11
2018/7/31

カーネルテンプレート化と計算再利用による CNNの計算量削減に関する検討

井内 悠太[1]　津邑 公暁[1]

概要：画像認識において，Convolutional Neural Network (CNN) と呼ばれるニューラルネットワークが高い認識精度を示し広く利用されている．近年では，CNN が学習や推論に要する計算量の増大が問題となっており，この計算量を削減するための研究が盛んに行われている．しかし，計算量を削減する方法によっては，CNN の認識精度が大きく低下する場合も存在し，CNN の認識精度の低下を抑えつつ，計算量を削減する事が課題となっている．本稿では，学習のための計算量削減に向けたカーネルテンプレート化と，推論のための計算量削減に向けた計算再利用の活用について検討する．評価の結果，カーネルテンプレート化により，認識精度を損うことなく，学習させる必要のあるパラメタを大きく削減できることを確認した．また，CNN に含まれる処理において，繰り返し同じ計算が行われることを明らかにし，計算再利用が CNN 計算に対して高い親和性を持つことを確認した．

1. はじめに

画像認識において，**Convolutional Neural Network (CNN)** と呼ばれるニューラルネットワークが高い認識精度を示し，顔認識や文字認識，自動運転における歩行者認識などのアプリケーションに広く利用されている．一方で，CNN の学習には膨大な計算資源と時間コストが必要なことが知られており，この計算コストの大きさが問題となっている．近年では認識精度を向上させるために，CNN の規模を増大させる傾向にあり，これに伴って，必要な計算時間や消費エネルギーも更に増大を続けている．

この問題に対し，CNN に含まれる計算を近似することで計算量を削減する研究が盛んに行われている．しかし，近似方法によっては元のデータが持つ情報が大きく損なわれ，CNN の認識精度が大きく低下してしまう．このような認識精度の低下は，アプリケーションによっては無視できない場合も多い．

そこで本研究では，CNN の認識精度の低下を抑えつつ，学習に要する時間，および，推論に要する時間の両方を削減する方法について検討する．学習時間削減のために，カーネルをテンプレート化するという新しい視点を提案し，この効果を評価する．また推論時間削減のために，CNN 計算に対する計算再利用の親和性について議論・評価する．

[1] 名古屋工業大学
Nagoya Institute of Technology

2. CNN

CNN とは，畳み込み層と呼ばれる特別な層を持つニューラルネットワークである．畳み込み層は，**カーネル**と呼ばれる二次元フィルタ状のパラメタを複数持ち，**マップ**と呼ばれる二次元画像を入力として受け取り，このマップにカーネルを畳み込むことで得られた新たなマップを出力する．これにより画像の特徴を抽出することで，高い認識精度を達成している．

畳み込み層では，入力マップに対して複数のカーネルを畳み込む処理を行う．畳み込み処理とは，入力マップに対してカーネルを重ね合わせ，重なった値同士の積を求め，得られた積の総和を計算することで出力マップの各要素値を求める処理である．畳み込み層における，入力マップに対して複数のカーネルを畳み込む処理について，**図 1** に示す例を用いて説明する．この図は，ある畳み込み層において，横 X 個，縦 Y 個の要素で構成される入力マップに対して，$A \times B$ 個の要素から構成される F 枚のカーネルを畳み込むことで，出力マップを計算する様子を表している．図中の例では，入力マップの左上にカーネルを重ね合わせ，出力マップの左上の要素の値を求めている．この要素を計算した後，入力マップ上でカーネルを右にスライドし，同様の計算をすることで出力マップの別の要素の値を求める．これを繰り返していくことで，出力マップの全ての要素の値を計算する．

なお，畳み込み層が持つパラメタであるカーネルの各要

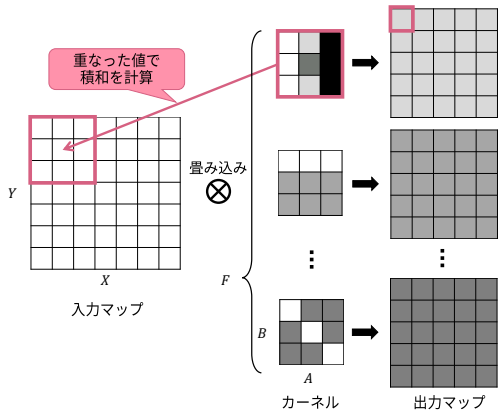

図 1 畳み込み層における計算

素値は，CNN が十分に高い認識精度を得られるように，**学習**と呼ばれる処理を事前に行うことで適切な値に調節されている．この学習処理では，CNN に画像データを入力して出力を求め，この出力と正解の出力との差に基づいてパラメタを更新する処理を繰り返し行う．また，学習後の CNN に画像データを入力し，その画像の認識結果を出力させる処理を**推論**と呼ぶ．

学習および推論のどちらにも共通して，畳み込み層の計算に多くの積和演算が必要であり，この計算コストの削減が CNN における課題となっている．さらに，近年では認識精度を向上させるために CNN の層数，カーネル数を増加させる傾向があり，これに伴って，学習時および推論時に必要な計算量およびデータ量が増大している．例えば，2012 年に発表された AlexNet[1] が持つ畳み込み層におけるパラメタ数および積和演算数はそれぞれ，230 万，6 億 6600 万であるのに対し，2014 年に発表された VGG-16[2] ではそれぞれ，1470 万，153 億と大幅に増加している．このような大規模な CNN においては，学習および推論のどちらにおいても処理に要する時間が長くなってしまうという問題がある．例えば，2015 年に発表された ResNet[3] は 152 層と層が深く，ImageNet データセット [4] を用いた学習には，最新の GPU を複数搭載した計算機を用いても，数週間を要することが報告されている．また，AlexNet に 1 枚の画像を入力し，GPU を用いて出力を計算するのに要する時間は 0.54ms である一方，VGG-16 の出力計算に要する時間は 10.67ms であることが報告されている [5]．このような計算時間の増加は，CNN を実用的なアプリケーションに応用する際に問題となる．CNN はリアルタイム性が求められるアプリケーションに応用されることも多く，そのような場合には計算時間が大きくなることで実用性が低下してしまうと考えられる．そのため，CNN の認識精度を高めつつ，CNN の出力を高速に計算することが求められている．

3. 関連研究

近年，CNN は劇的に発展し続けており，AlexNet[1]，VGG-16[2]，ResNet[3] など様々なアーキテクチャが提案されている．この発展により，CNN の認識精度が向上し続けているが，一方で，CNN の大規模化による計算コストの増加が問題となっている．この問題に対し，CNN に必要な計算量を削減するため，様々な研究が行われている．

3.1 計算量削減に関する既存研究

認識精度を低下させることなく CNN に必要な計算量を削減することを目指し，様々な研究が行われている．計算量の削減に向けた一つのアプローチとして，パラメタを量子化する方法が挙げられる．例えば，XNOR-NET[6] では入力や重みを，+1 または −1 の 2 値に限定することで，CNN 内の乗算を単純な XNOR 演算で置き換えている．この量子化によって，CNN パラメタのデータサイズを 1/32 に縮小でき，推論処理を 52 倍高速に実行できることが確認されている．また，重みを二進対数で表現することで，乗算をシフト演算に置き換える研究も行われている [7]．

しかし，以上で述べたようなパラメタの量子化は，学習によって獲得したパラメタを，量子化方針に沿うような近い数値に丸めるため，CNN の認識精度が有意に低下してしまう場合が多い．アプリケーションによってはこれが無視できなくなる可能性がある．

また，量子化による計算量削減とは異なるアプローチとして，ハードウェアアクセラレータの利用が挙げられる．例えば．Chen らは，大規模な CNN における推論処理を高速に行うアクセラレータである DianNao[8] を開発している．このアクセラレータは，マップを格納するための入力バッファ，出力バッファ，重みバッファ，および多数の積和演算器を備えた NFU（Neural Functional Unit）から構成されている．このように，多数の積和を並列に計算可能な演算器を備えたコアと，データ移動が少なくなるように入出力マップ，およびカーネル用のバッファを用意しておくことで，CNN における推論処理の高速化を実現している．しかし，近年の大規模化する CNN の出力を高速に計算するためには，同時に計算可能な積和の数をさらに増加させる必要がある．そのため，このようなハードウェアアクセラレータの計算速度を追求すると，コア数などを増加させなければならず，回路面積や消費電力も増大してしまう．

これらの研究に対し我々は，異なるデータセットを用いて学習した異なる CNN モデル間にも類似したカーネルが存在する [9] ことに着目し，CNN の認識精度の低下を抑えつつ計算量を削減可能な，高電力効率なアクセラレータを提案している [10]．当該研究では，まずカーネルをクラス

タリングし，クラスタ内のカーネルを，そのクラスタを代表するカーネルで置き換えて処理を共通化する．そして，この共通化した処理は，多くの CNN で行われる処理であると考えられるため，この処理に特化したハードウェアを用意することで，推論を高速化した．

この先行研究により得られた，「様々な CNN 間で似た機能を持つカーネルが存在する」という知見と，畳み込み層ではエッジやブロブなどの単純な特徴を抽出する機能が必要という予見から，カーネルの一部をテンプレート化できるのではないかという着想を得た．そこで本稿では，このカーネルテンプレート化が，学習・推論時間に与える影響と，認識精度等に与える影響とを評価する．

3.2 Local Binary Convolutional Neural Networks

この我々の発想に近い考え方を導入した既存研究に，Juefei-Xu らが提案する Local Binary Convolution（LBC）[11] がある．当該研究では，標準的な CNN における畳み込み層を LBC 層で代替することで，計算量削減を実現している．LBC 層は，学習中に更新されない，予め設定された二値のパラメタを持つスパースなカーネルのセットと，非線形活性化関数と学習により重みが更新される全結合層とのセットとで構成される．なお，標準的な CNN における畳み込み層を LBC 層で代替した CNN は，Local Binary Convolutional Neural Networks（LBCNN）と呼ばれる．LBCNN の学習においては，カーネルのパラメタを最適化するのではなく，全結合層の重みのみを最適化することとなる．そのため，LBC 層は，標準的な CNN 層と比較して，学習させるパラメタ数を大幅に削減可能である．

LBC 層におけるパラメタは，Local Binary Patterns（LBP）の考えに基づいて設定される．LBP とは，顔認識分野から生まれたシンプルかつ強力な特徴量であり，パターン認識や画像処理アプリケーションなどで広く用いられている．ここで，LBP 特徴量の算出方法を，**図 2** に示す．LBP 特徴量を算出する際，まず画像の一部をパッチとして切り出し（図 2 左，中），そのパッチにおける中心画素の輝度と，その周辺画素の輝度とを比較する．このとき，周辺画素の輝度が中心画素の輝度よりも大きければ 1，小さければ 0 とする（図 2 右）．そして，この比較した結果が，パッチおける中心画素の LBP 特徴量となる．当該研究では，この考えを応用し，指定されたスパースレベルに応じて，カーネルのパラメタを 0，あるいは 1 か −1 のランダムな二値に設定する．そして，このランダムに生成されたカーネルを，LBC 層におけるカーネルとして用いる．

LBCNN は非常にシンプルなモデルであるにもかかわらず，過学習が起こりにくく，リソースが制約された環境における学習および推論に適していることが示されている．また，LBC 層が CNN 層の良い近似であることが，理論的に，かつ経験的に証明されている．LBCNN は，標準的な

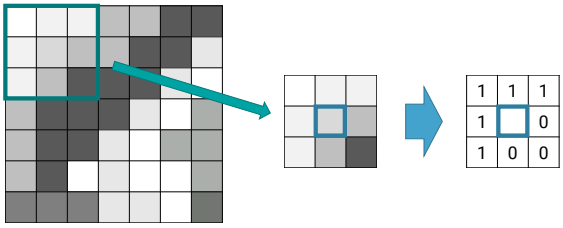

図 2 LBP 特徴量

CNN と比較して計算量を大幅に削減しながら，MNIST，SVHN，CIFAR-10，ImageNet データセットを用いた評価では通常の CNN と同等の性能を達成している．

我々はこの考え方を更に進め，アプリケーションの違いによらず必要となるであろう，エッジやブロブなどの単純ではあるが意味のある特徴を抽出するカーネルを，テンプレートのような形で予め用意し，ネットワーク内に埋めこむことで，CNN 学習の際の処理量削減が期待できるのではないかと考えた．なお，これらテンプレートに用いるパラメタは，4.2 節で後述するように二値で表現するため，同じ値による乗算が畳み込み演算中に多数現れ得る．このような同一の乗算を計算再利用により省略することで，推論の処理量も削減できる可能性がある．本稿では，これらアイディアの詳細と，期待できる効果に関する調査結果について示す．

4. カーネルテンプレート化

本章では，カーネルテンプレート化の概要，およびカーネルテンプレート化による学習対象パラメータ量の削減について述べる．

4.1 カーネルテンプレート化の概要

学習済み CNN では一般に，畳み込み層においては，浅い層で単純な特徴が抽出され，層が深くなるに従って，抽出される情報が抽象化されていくことが知られている [12]．生物の脳でも，一次視覚野の一部の領域が，特定方向のエッジなどに反応し認識する機能を持つことが知られているが，これと同様，CNN の浅い層においては，エッジやブロブといった単純な特徴の検出を担っているのではないかと考えられるカーネル（**図 3**）が学習によって出現する場合が多い．

このようなカーネルが実際にエッジやブロブを検出する目的で存在しているのだとすると，それらは学習を通じて得られたものであるがゆえ，最初からそのような目的で設計したカーネル（**図 4**）と比較すると，ノイズとも言えるわずかな値のブレが含まれる．我々は，このような値のブレは本来必要のないものであると考え，いずれのアプリケーションにも必要となるであろう単純な特徴抽出に特化したカーネルをテンプレートとして用意し，これらを構成するパラメータを学習対象から除外して固定することで，学習

図 3　浅い層に現れるカーネル

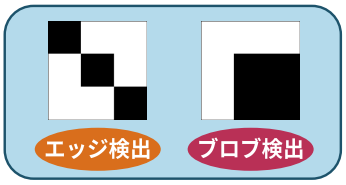

図 4　特徴検出に特化したカーネル

初期段階からの高い認識率と，学習に要する処理量削減につながるのではないかと考えた．

本稿では，このような単純な特徴抽出に特化し，学習対象から除外してパラメータを固定したカーネルを，**テンプレートカーネル**と呼ぶ．また，畳み込み層を構成する層に含まれるカーネルの一部，あるいは全てをテンプレートカーネルとして固定することを，カーネルテンプレート化と呼ぶ．

ニューラルネットワークが持つパラメタは，学習によって最適な値に近付くように更新される．ここで，最適な値というのは，損失関数の値を最小にする値のことである．また，学習させる際には，一般的に勾配降下法が用いられる．勾配降下法を用いた更新処理では，各層ごとに，その層すべてのパラメタに関する損失関数の勾配，すなわちそれぞれのパラメタに関する偏微分を求める必要があり，これが学習時の計算コストが大きくなる一因となっている．カーネルテンプレート化により一部のパラメタを固定することで，その固定したパラメタに対する更新処理が不要となるため，更新処理を省略することができる．

4.2　評価

カーネルテンプレート化が CNN の認識精度および計算量に及ぼす影響について評価した．ディープラーニングフレームワークである Chainer[13] 上で評価を行い，MNIST データセット [14] と CIFAR-10 データセット [15] とを用いて CNN を学習させた．MNIST データセットを学習させた CNN モデルを**図 5** に，CIFAR-10 データセットを学習させた CNN モデルを**図 6** に示す．

なおテンプレートカーネルとしては，高次局所自己相関特徴（Higher-order Local Auto-Correlation: HLAC）[16] に基づいて決定した**図 7** に示す 25 枚のパターンを用い，畳み込み層の第 1 層の一部をこの 25 枚のテンプレートカーネルで固定した．HLAC は，画像認識のための統計的特徴量であり，位置不変性や加法性を満たすという特徴を持つ．

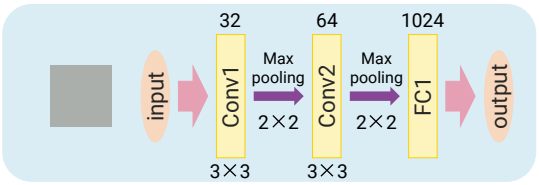

図 5　MNIST データセット用 CNN モデル

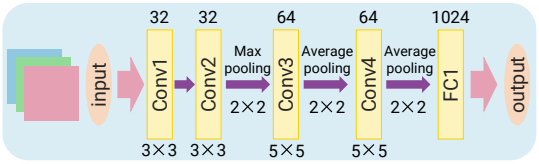

図 6　CIFAR-10 データセット用 CNN モデル

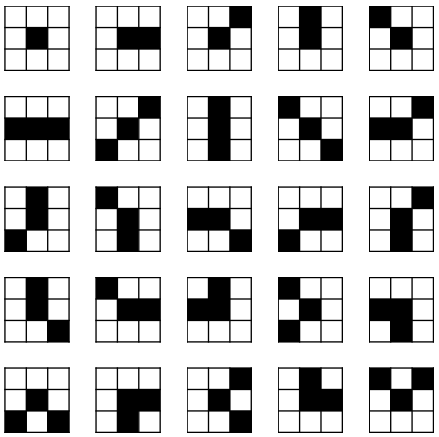

図 7　HLAC に基づいて決定したテンプレートカーネル

図 7 のテンプレートカーネルのもととなった，HLAC に基づくマスクパターンでは，各要素値（相関値）は $\{0, 1\}$ の二値で表されるため，テンプレートカーネルの要素値も二値とする．ここで，Chainer がランダム生成する初期値は，平均 0，分散 $1/\sqrt{fan\_in}$（fan_in：入力数）のガウス分布に従うことから，理論的には約 99.7% が $[-3/\sqrt{fan\_in},\ 3/\sqrt{fan\_in}\,]$ の範囲に収まる．そのため，図 7 に示すテンプレートカーネルにおいて，黒で示されている要素値を $3/\sqrt{fan\_in}$，白で示されている要素値を $-3/\sqrt{fan\_in}$ とした二値で表すこととした．

認識精度の評価結果を**図 8** と**図 9** に，計算量の評価結果を**表 1** 示す．図 8 は MNIST データセットを用いて学習させた際の認識精度を，図 9 は CIFAR-10 データセットを用いて学習させた際の認識精度を，それぞれ示している．それぞれの図において，グラフの縦軸は認識精度，横軸はエポック数を表している．評価結果は 4 本の折れ線グラフで表されており，橙色がカーネルテンプレート化を適用した場合の認識精度を，また，緑色がカーネルテンプレート化を適用していない場合の認識精度を，それぞれ示している．なお，実線が Top1-Accuracy を，破線が Top5-Accuracy を，それぞれ表している．表 1 は，カーネルテンプレート化を適用する前後それぞれに場合における，CNN の一層

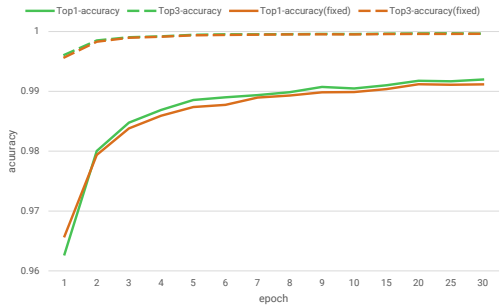

図 8 認識精度（MNIST データセット）

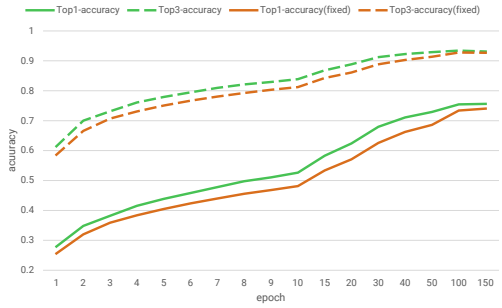

図 9 認識精度（CIFAR-10 データセット）

表 1 学習対象となるパラメタ数（一層目）

カーネルテンプレート化適用前	864
カーネルテンプレート化適用後	189

目において学習させる必要のあるパラメタの数を示している．

図 8，図 9 において，カーネルテンプレート化を適用した場合と適用していない場合それぞれの認識精度を比較すると，学習後においてはカーネルテンプレート化を適用した場合の方がやや認識精度が低下しているが，それほど大きな差はないことが確認できる．一方で表 1 より，CNN の一層目において学習させる必要のあるパラメタの数は，カーネルテンプレート化により 864 から 189 まで，約 78% 削減できることが確認できる．したがって，カーネルテンプレート化は，認識精度を損うことなく計算量を削減するために有効だと考えられる．

5. CNN 推論処理への計算再利用の応用

本章では，CNN 推論処理に対する計算再利用の親和性について述べ，特に前章までで述べたカーネルテンプレート化を適用して場合における，推論処理の削減量について見積りを行う．

5.1 計算再利用による推論処理の削減

計算再利用とは，計算に用いた入力と出力の組を記憶しておき，同じ入力による再計算が必要となった際，記憶しておいた出力を再利用することで再計算自体を省略し，高速化を図る手法である．元来，計算再利用は，関数型言語などにおいて，再帰関数のような重い関数に対して効果を発揮するプログラミングテクニックであったが，ハードウェア的にこれをサポートすることでプロセッサ高速化にも活用する研究がなされている．それらの中には，単命令に対して，オペランドが過去と同一であった場合に実行を省略するといった細粒度なものから，関数やループボディの処理をまとめて省略するものまで，幅広く研究されている．本稿では，CNN における処理に対する計算再利用の適用による推論処理量の削減について検討する．

冒頭でも述べたように，CNN は画像認識などに広く用いられているが，一般に画像中のある画素の画素値と，その近傍画素の画素値とは，近い値をとる傾向にあることが知られている [17]．それゆえ，CNN の入力画像を構成する画素値，ひいては，入力マップを構成する要素値にも偏りがあると考えられる．また，4.1 節で述べたカーネルテンプレート化を CNN に対して適用した場合，テンプレートカーネルを構成する要素値は 2 種類となるため，畳み込み処理には，入力マップの要素値とカーネルのパラメタ値とが同じ組み合わせとなる乗算が多く含まれ，計算再利用と親和性が高いと考えられる．

そこで，推論処理に対して計算再利用が有効となりうるかを確認するため，乗数・被乗数の組み合わせが同じであるような乗算が推論処理中にどの程度出現するかについて調査した．

表 2 乗数・被乗数の組合せ数（カーネルテンプレート化適用前）

出現回数/精度	fixed8	float16	float32
1〜5	14215.2	96793.6	102700.0
6〜10	8101.7	28584.9	29299.1
11〜50	16622.8	15218.4	14451.2
51〜100	2453.2	546.7	540.7
101〜500	686.2	241.2	233.6
501〜1000	14.9	0.0	0.0
1001〜5000	0.0	0.0	0.0
5001〜10000	0.0	0.0	0.0
10000〜	0.0	0.0	0.0
組合せ総数	42094.0	141384.8	147224.6

表 3 乗数・被乗数の組合せ数（カーネルテンプレート化適用後）

出現回数/精度	fixed8	float16	float32
1〜5	11958.5	22390.0	22659.7
6〜10	5698.2	8117.4	8155.7
11〜50	5017.7	3269.6	3225.7
51〜100	71.1	11.4	11.0
101〜500	91.9	89.7	89.7
501〜1000	102.4	102.1	102.1
1001〜5000	227.8	228.0	228.0
5001〜10000	4.9	4.9	4.9
10000〜	0.3	0.3	0.3
組合せ総数	23172.8	34213.4	34477.1

5.2 調査結果

図 6 に示すモデルの一層目における畳み込み演算について，カーネルテンプレート化を適用しない場合と適用した場合それぞれにおいて，乗数・被乗数の組み合わせが同じであるような乗算がどの程度出現するかを調査した．なお一層目において 1 ステップで実行される乗算の総回数は，ノード数 32，ノード内チャネル数 3，各チャネルにおけるの畳み込み演算回数 30×30，一度の畳み込みで実行される乗算数 3×3 より，777,600 回である．3.1 節で述べたように，CNN ではパラメタの量子化が適用されることが多いため，パラメタを Chainer 標準である 32bit 浮動小数点で表現した場合に加え，16bit 浮動小数点，8bit 固定小数点で量子化した場合についても調査した．調査結果を，**表 2** および**表 3** に示す．

これらの表は，畳込み処理で行われる乗算において，乗数・被乗数の組合せが何組出現したかを，同じ計算が行われた回数ごとに分類して示したものであり，8bit 固定小数点に量子化した場合（fixed8），16bit 浮動小数点に量子化した場合（float16），量子化を行わない場合（float32）のそれぞれの結果をまとめている．なお表に示した値は，入力データセットからランダムに 10 個を選択し，それぞれに対して推論を行う際に現れる乗算について調査した結果の平均値である．

例えば表 2 において，float32 の列の 51 ～ 100 の行に 540.7 という値が記されている．これは，量子化を行わない場合，図 6 に示した CNN の一層目の畳み込み層で行われる乗算において，約 540 組の乗数・被乗数ペアに対して，それぞれ 51 ～ 100 回同じ乗算が行われたということを表している．

表 2 から，8bit 固定小数点に量子化した場合は，出現回数が 50 回を超える乗数・被乗数の組み合わせが多く存在しており，同じ乗数・被乗数の組み合わせによる乗算が数多く繰り返し実行されていることが確認できる．しかし，量子化の有無にかかわらず，出現回数が 500 回を越えるような組はほとんど存在しておらず，計算再利用を用いても省略できる乗算は限定的であると考えられる．

一方で，表 3 に示す，カーネルテンプレート化を適用した場合の結果からは，適用前に比べて多くの計算が重複していることが見てとれ，数千から 1 万回出現する組まで存在していることから，計算再利用による計算量削減が有効に働くであろうことが予想できる．例えばカーネルテンプレート化を適用せず，量子化も適用しない場合，重複していない乗算のパターンは約 147,224 個あるため，計算再利用によって乗算回数が約 19%（= 147224/777600）まで削減できるが，カーネルテンプレート化を適用した場合では，これが約 4.4%まで削減でき，更に 8bit 固定小数点に量子化した場合では，乗算回数を約 3.0%まで削減可能であることが分かる．

6. おわりに

本稿では，CNN の学習に要する処理量削減を目的としたカーネルテンプレート化を提案し，また CNN の推論に要する処理量削減を目的として，CNN 計算に対する計算再利用の活用の可能性について議論した．カーネルテンプレート化により，認識精度を損うことなく，学習対象となるパラメタ数を大きく削減できることを確認した．また，CNN 計算における乗算については，繰り返し出現する乗数・被乗数の組合せが数多く存在することを確認し，計算再利用が CNN 計算に対して高い親和性を持つことが確認できた．更に，カーネルテンプレート化の適用により，より多くの乗数・被乗数の組合せが繰り返し出現することが確認でき，計算再利用の効果が更に大きくなることが予測できた．今後の課題としては，より汎用的なテンプレートカーネルの設計，および，計算再利用を適用する具体的な方法の考案などが挙げられる．

参考文献

[1] Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks, *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pp. 1097–1105 (2012).
[2] Simonyan, K. and Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556 [cs.CV] (2014).
[3] He, K., Zhang, X., Ren, S. and Sun, J.: Deep Residual Learning for Image Recognition, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2016).
[4] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge, *Int'l Journal of Computer Vision (IJCV)*, Vol. 115, No. 3, pp. 211–252 (online), DOI: 10.1007/s11263-015-0816-y (2015).
[5] Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L. and Shin, D.: Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications, arXiv:1511.06530 [cs.CV].
[6] Rastegari, M., Ordonez, V., Redmon, J. and Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks, *European Conference on Computer Vision*, Springer, pp. 525–542 (2016).
[7] Miyashita, D., Lee, E. H. and Murmann, B.: Convolutional Neural Networks using Logarithmic Data Representation, arXiv:1603.01025 [cs.NE] (2016).
[8] Chen, T. et al.: DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning, *Proc. 19th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS'14)*, pp. 269–284 (2014).
[9] Denton, E., Zaremba, W., Bruna, J., LeCun, Y. and Fergus, R.: Exploiting Linear Structure within Convolutional Networks for Efficient Evaluation, *Proc. 27th Int'l Conf. on Neural Information Processing Systems (NIPS'14)*, Vol. 1, pp. 1269–1277 (2014).
[10] 進藤智司, 松井優樹, 八巻隼人, 津邑公暁, 三輪　忍：カー

ネルの類似性に基づく近似計算を行う CNN アクセラレータの検討，情処研報 (ETNET2018)，Vol. 2018-ARC-230, No. 31, pp. 1–6 (2018).
[11] Juefei-Xu, F., Boddeti, V. N. and Savvides, M.: Local Binary Convolutional Neural Networks, arXiv:1608.06049 [cs.LG] (2016).
[12] Mahendran, A. and Vedaldi, A.: Understanding Deep Image Representations by Inverting Them, *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5188–5196 (2015).
[13] Tokui, S., Oono, K., Hido, S. and Clayton, J.: Chainer: A Next-Generation Open Source Framework for Deep Learning, *Proc. of Workshop on Machine Learning Systems (LearningSys) in the 29th Annual Conf. on Neural Information Processing Systems (NIPS)* (2015).
[14] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P.: Gradient-based Learning Applied to Document Recognition, *Proc. of the IEEE*, Vol. 86, pp. 2278–2324 (1998).
[15] Krizhevsky, A. and Hinton, G.: Learning Multiple Layers of Features from Tiny Images, Technical report, Univ. of Toronto (2009).
[16] Otsu, N. and Kurita, T.: A New Scheme for Practical Flexible and Intelligent Vision Systems, *Proc. IAPR Workshop on Computer Vision*, pp. 431–435 (1988).
[17] Miguel, J. S., Albericio, J., Jerger, N. E. and Jaleel, A.: The Bunker Cache for spatio-value approximation, *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12 (online), DOI: 10.1109/MICRO.2016.7783746 (2016).