Vol.2018-ARC-230 No.31
Vol.2018-SLDM-183 No.31
Vol.2018-EMB-47 No.31
2018/3/8

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

進藤 智司[1]　松井 優樹[2]　八巻 隼人[2]　津邑 公暁[1]　三輪 忍[2]

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，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があり，これに伴って，計算時間や消費エネルギーも増大している．この問題に対して，CNN の計算に特化したアクセラレータが開発されている．その中でも，CNN に含まれる計算を近似することで，電力効率の向上を図る研究が盛んに行われている．しかし，CNN の認識精度の低下を抑えつつ，高い電力効率を持つアクセラレータは未だ実現されていない．そこで本研究では，近似計算をアクセレラレータに導入し，認識精度の低下を抑えつつ高電力効率なアクセラレータの実現を目指す．特に，様々な CNN に共通して含まれる代表的なカーネルに特化した回路を予め備えたアクセラレータを検討する．

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．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があり，これに伴って，計算時間や消費エネルギーも増大している．この問題に対し，CNN の計算に特化したハードウェアアクセラレータが多数開発されている．その中でも，CNN に必要となる多量の積和演算に着目し，積和を並列に計算可能な構造を持つアクセラレータが多く提案されている [1], [2]．これらのアクセラレータは，GPU と比較して消費エネルギーを約 1/100 に抑えることが可能であるが，CNN の大規模化に対応するためには，並列度をさらに高める必要があり，回路面積が増大してしまう．

一方，CNN に含まれる計算を近似することで計算量を削減し，単純な回路で構成する CNN アクセラレータが研究されている．しかし，近似方法によっては元のデータの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く，CNN の認識精度の低下を抑えつつ，高い電力効率を持つアクセラレータの実現が望まれている．

そこで本研究では，CNN の認識精度の低下を抑えつつ高電力効率なアクセラレータの実現を目指す．特に，様々な CNN に共通して含まれる代表的なカーネルに着目し，このカーネルに特化した計算ユニットを備えたアクセラレータを検討する．このアクセラレータが行う近似計算が認識精度へ与える影響や，アクセラレータの計算性能を調査することで，実現性を確認する．

[1] 名古屋工業大学
Nagoya Institute of Technology
[2] 電気通信大学
UEC

2. 研究背景

本章では，CNN の概要および CNN を実用的なアプリケーションに用いる際の問題点について述べる．

2.1 CNN の概要

CNN は畳み込み層と呼ばれる特別な層を持ち，これによって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．この畳み込み層は，**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つ．また，畳み込み層は**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．

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っている．この計算を**図 1** に示す例を用いて説明する．この図は，ある畳込み層において，入力マップに対して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．入力マップは横 X 個，縦 Y

Vol.2018-ARC-230 No.31
Vol.2018-SLDM-183 No.31
Vol.2018-EMB-47 No.31
2018/3/8

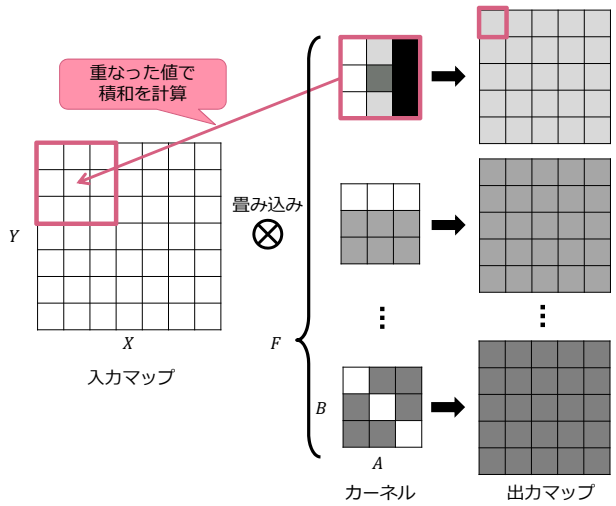

図 1　畳み込み層における計算

個の要素で構成され，これに対して $A \times B$ 個の要素から構成された F 枚のカーネルを畳み込んでいる．畳み込み演算は，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．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，また同様の計算によって出力マップの 1 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．このような処理によって 1 枚のカーネルを畳み込むが，この畳み込み層に含まれる F 枚のカーネルは独立しており，カーネル間にデータの依存関係はないため，複数カーネルの畳み込み演算は同時に計算することができる．

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，この出力と理想的な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，CNN の出力を求める処理が含まれているため，本研究では CNN の出力計算に着目する．

2.2　問題点

CNN が持つ畳み込み層の出力を求めるには多数の積和を計算する必要がある．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，これに伴って出力計算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[3] が持つ畳み込み層のパラメタ量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[4] ではそれぞれ，1470 万，153 億と大幅に増加している．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，計算時間が大きくなることで実用性が低下してしまう．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

本章では，CNN に必要な計算を高速かつ低消費電力に行うことを目指した既存研究について述べる．特に，パラメタを近似することで計算量を削減する方法と，アクセラレータを利用することで計算を高速化する方法の 2 つのアプローチについて述べる．

3.1　パラメタの近似による計算量削減

認識精度を低下させることなくパラメタを近似し，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．パラメタを近似することで計算量を削減する方法の 1 つに量子化がある．例えば，XNOR-NET[6] では入力や重みを，+1 または −1 の 2 値に近似することで，CNN 内の乗算を単純な XNOR 演算で置き換えている．この近似化によって，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，重みを二進対数に変換することで，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近似は，学習によって獲得したパラメタに変更を加えるため，近似方法によっては CNN の認識精度が有意に低下してしまい，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．そのため，認識精度の低下を抑えられるように，適切にパラメタを近似する必要がある．

3.2　CNN アクセラレータ

CNN の出力計算を高速化することを目指した既存研究のうち，パラメタの近似等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[1] 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

Vol.2018-ARC-230 No.31
Vol.2018-SLDM-183 No.31
Vol.2018-EMB-47 No.31
2018/3/8

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 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ない．コアには積和演算器が多数含まれるため，回路面積や消費電力も増大してしまう．

そこで，先ほど述べたパラメタの近似を応用することで，積和演算器よりも単純な回路でアクセラレータを構成する研究も行われている．Nakahara ら [8] は，CNN 内の float 型で表されたデータを，+1 または −1 の 2 値で扱うアクセラレータを提案している．入出力やカーネルを 2 値で扱うことで，回路面積，消費電力の大きい乗算器を，よりコストの小さい XNOR 演算器で置き換えている．しかし，このような 2 値への近似を施すと，パラメタが持つ情報が大きく損なわれるため，CNN の認識精度が大きく低下してしまう．このように，CNN の認識精度を保ちつつ，高効率に CNN の出力を計算するアクセラレータはいまだ実現されていない．そこで本研究では，CNN の認識精度の低下を抑えつつ計算量を削減可能な計算の近似方法を検討する．そして，その計算の近似をアクセラレータに導入し，認識精度の低下を抑えつつ高電力効率なアクセラレータの実現を目指す．

4. カーネルの類似性に基づく近似を応用した CNN アクセラレータ

本章では，検討するアクセラレータの設計方針および概要について述べる．

4.1 設計方針

一般的に，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ことが知られている．それらの類似したカーネルは，各要素値に多少の違いはあれど，CNN 内で果たす役割はほぼ等しく，機能面で同等のカーネルであると考えることができる．そこで，これら同等の機能を持つ複数のカーネルを同一のカーネルと見なし，畳み込み演算を共通化することで計算量を削減する手法が既に提案されている [9]．

これに対し，本研究ではさらに，類似したカーネルの中でも特に，様々な CNN に共通して多く含まれる代表的なカーネルに着目する．例えば，“特定方向のエッジを抽出するパターンを持つカーネル”，“特定の色成分を抽出するパターンを持つカーネル” など，入力データから単純なパターンを抽出する機能を持つカーネルは，あらゆる CNN に共通して含まれていると考えられる．この代表的なカーネルを**コモンカーネル**と定義し，コモンカーネルの畳込み演算を高効率に行うことに特化した回路を予め備えたアクセラレータを実現する．本稿では，アクセラレータに搭載するこの回路を**専用計算ユニット**と呼ぶ．このアクセラレータを用いることで，既に様々なアプリケーションで使用されている CNN はもちろんのこと，今後新たに開発される可能性のある CNN モデルを含む，あらゆる CNN の出力計算を効率化することができると考えられる．

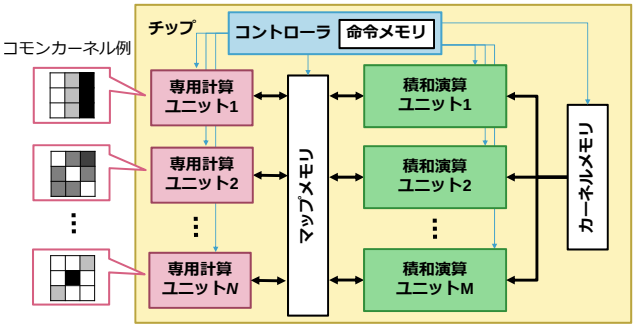

図 2　アクセラレータの全体像

4.2 アクセラレータの全体像

以上で述べた方針にしたがって設計するアクセラレータの全体像を図 2 に示す．アクセラレータは，マップメモリとカーネルメモリ，複数の汎用の積和演算ユニットと専用計算ユニット，それらを制御するコントローラと命令メモリから構成される．専用計算ユニットはそれぞれ異なるコモンカーネルを基に設計されており，各コモンカーネルとの類似度が高いカーネルの畳み込み演算を担う．また，積和演算ユニットは専用計算ユニット群で計算することができないカーネルの計算，および畳み込み層以外の層の計算を行う．カーネルの畳み込み演算を行う際，積和演算ユニットはカーネルデータを必要とするため，カーネルデータを格納するためのカーネルメモリを積和演算ユニットに接続する．マップメモリは専用計算ユニットと積和演算ユニットのいずれにも接続しており，このマップメモリを介してマップデータを共有する．

5. CNN アクセラレータの実装

4 章で述べたアクセラレータを実現するために，コモンカーネルの決定方法と，コモンカーネルによる畳込みを高効率に計算する方法，の 2 点を検討する必要がある。本章では，この 2 点を決定する方法について述べる．

5.1 カーネルクラスタリングによるコモンカーネルの決定

アクセラレータに搭載する専用計算ユニットで計算できるコモンカーネルは，様々な CNN に広く含まれるカーネルを代表するものであることが望ましい．そこで，様々な CNN のカーネルをクラスタリングによって，類似した特徴を持つカーネルに分類し，そのクラスタリング結果を基に，代表的なカーネルを抽出する．特にカーネルが持つ機

Vol.2018-ARC-230 No.31
Vol.2018-SLDM-183 No.31
Vol.2018-EMB-47 No.31
2018/3/8

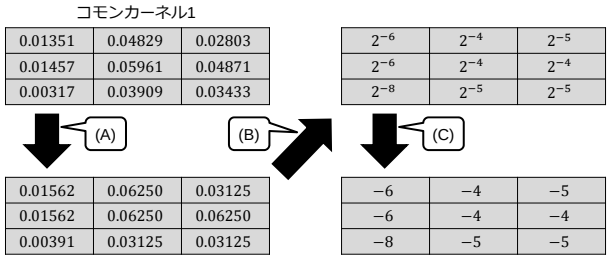

図 3 2 の冪乗値への丸め

能や，表しているパターンの類似性に着目し，1 枚のカーネルの全要素値をひとまとまりのデータとしてクラスタリングアルゴリズムに与えてクラスタリングを行う．その結果，類似した特徴を持つカーネルが同クラスタにまとめられるが，含まれているカーネルの数はクラスタによってそれぞれ異なり，さらにカーネルの各要素の絶対値の大きさも異なる．そのため，どのクラスタに含まれるカーネルをコモンカーネルとするかによって，専用計算ユニット群でカバー可能なカーネルの数や，認識精度の低下量が異なる．そこで，クラスタに対して一定の基準を設け，その基準を満たす各クラスタの代表のカーネルをコモンカーネルとする．なおこの代表のカーネルは，クラスタに含まれるカーネルの特徴を適切に表していることが望ましいため，クラスタに含まれるカーネルの要素ごとに平均値を求め，それを要素値として持つコモンカーネルを設計する．

5.2 2 の冪乗値への丸めによる近似

専用計算ユニットは，入力値とコモンカーネルの各要素どうしの乗算を効率的に行う必要がある．そのために，コモンカーネルの要素値を近似することで，乗算によって得られる結果をよりコストの小さい演算で近似的に求めることを考える．本研究では，コモンカーネルの各要素を最も近い 2 の冪乗値で近似する方法を採用する．これにより，カーネルの持つ情報が大きく失われることを防ぎつつ，乗算よりもコストの小さいシフト演算を使用可能にする．

ここで，コモンカーネルの各要素値を 2 の冪乗値へ丸める様子を**図 3** に示す．この図中の 3×3 の要素を持つコモンカーネル 1 を近似する場合について考える．まず，float 型の各要素値を最も近い 2 の冪乗値に置き換える（A）．例えば，1 行 1 列目の要素値は 0.1351 であるが，これに最も近い 2 の冪乗値を求めると 0.01562 となる．この 2 の冪乗値を，冪指数を p として 2^p の形で表現し（B），各要素の冪指数 p を並べる（C）．なお，2 の冪乗値と入力値を乗算することは，冪指数 p だけ入力値をシフトすることと同意である．つまり，冪指数 p が正の場合は左シフト，負の場合は右にシフトすればよい．このような変換によって，入力と各要素値の乗算を，入力値のシフトで代替可能となる．

以上で述べた近似を用いて，専用計算ユニットを構成する．コモンカーネルの float 型の各要素値がシフト量に変換され，乗算をシフト演算で代替することができる．また，専用計算ユニットは 1 枚のコモンカーネルのみに特化しているため，各要素値のシフト幅は固定することができる．そのため，専用計算ユニットは，任意のビット数をシフト可能なシフタを必要としない．加算器の入力に対して，入力マップの値を固定量だけずらして配線しておくだけで，積和演算結果が得られる．さらに，コモンカーネルの各要素値はこの配線の「ずれ」によって回路中に埋め込まれているため，専用計算ユニット外のメモリからカーネルデータを読み出す必要がない．

6. 評価

以上で述べたアクセラレータの実現性を確認するために評価を行った．評価では，カーネルクラスタリングに基づいて設計したアクセラレータを用いて CNN の推論を行った場合の認識精度を調査し，その際のアクセラレータの性能を見積もる．

6.1 評価環境

6.1.1 クラスタリング方法

今回の評価では，アクセラレータを設計するために，VGG-16[4] に含まれるカーネルを基にコモンカーネルを決定する．この VGG-16 において，データセット ImageNet[10] を用いて学習処理を行った場合のカーネルデータを使用する．VGG-16 は 13 層の畳み込み層を持ち，合計で 1634496 枚のカーネルが含まれている．全てのカーネルのサイズは同じ 3×3 であり，これらをそれぞれ $3 \times 3 = 9$ 次元のデータとしてクラスタリングアルゴリズムに与える．クラスタリングアルゴリズムには K-means[11] を用い，128 クラスタに分類した．

6.1.2 認識精度の調査方法

アクセラレータを用いて近似的に CNN の出力を計算した際の認識精度への影響を調査するために，データセットとして ImageNet の画像と正解ラベルのセットを 1000 枚使用する．画像を VGG16 に入力し，出力を計算することでその画像に映る物体を 1000 個のクラスに識別させる．この識別結果は，各クラスにその物体が属する確率として出力される．本評価では，識別結果の第一候補，つまり確率が最も高いクラスが正解ラベルと一致する割合である Top1-accuracy，および，識別結果の第五候補までに正解ラベルが含まれる割合である Top5-accuracy をそれぞれ求めた．

また，専用計算ユニットの数と認識精度の関係を調査するために，搭載する専用計算ユニットの数を変化させた場合の認識精度を調査した．この際，コモンカーネルを生成するクラスタを選択する際の基準値として，クラスタサイズを用いた．これは，1 つのクラスタに含まれるカーネルの枚数が多いほど，そのクラスタを対象として生成したコ

Vol.2018-ARC-230 No.31
Vol.2018-SLDM-183 No.31
Vol.2018-EMB-47 No.31
2018/3/8

モンカーネルは，より多くのカーネルとの類似度が高いという考えに基づいている．そのため，含まれるカーネルの数が多いクラスタから順に，コモンカーネルの生成対象として選択する．このような方法で専用計算ユニットの数を変化させながら，VGG-16 における推論処理を行った場合の認識精度を調査した．また，コモンカーネルの各要素値を float 型のまま計算に用いる場合と，最も近い 2 の冪乗値で丸めた場合でそれぞれ認識精度を比較することで，2 の冪乗値による近似が認識精度へ与える影響も調査した．

6.1.3 計算時間の見積もり方法

本評価では，アクセラレータを用いて CNN の推論処理を近似的に行い，その実行に要する時間を見積もる．なお，専用計算ユニットの搭載数を変動させて評価を行うにあたり，アクセラレータ全体が要するエネルギーの条件を一定に保つため，まず専用計算ユニットと積和演算ユニットそれぞれに要するエネルギー比を定義する．その上で，専用計算ユニット数の増減に合わせて，全体のエネルギーが一定に保たれるよう，積和演算ユニット数も増減させて評価を行った．ここで，32bit 浮動小数点型の乗算命令の実行に必要なエネルギーは 3.7pJ，32bit 浮動小数点型の加算命令の実行に必要なエネルギーは 0.9pJ であることが報告されている [12]．この値を用いて，それぞれのユニットが 1 回の計算に要するエネルギーを求める．本評価で想定する積和演算ユニットは VGG16 のカーネルサイズに合わせ，9 個の乗算器と 10 個の加算器から構成される．つまり，積和演算ユニットが 1 回の計算に要するエネルギーは，$3.7 \times 9 + 0.9 \times 10 = 42.3$pJ となる．一方で，専用計算ユニットは加算器 10 個から構成される．つまり，専用計算ユニットの 1 回の計算に要するエネルギーは，$0.9 \times 10 = 9$pJ となる．したがって，積和演算ユニットを 1 個減らした際に，専用計算ユニットを 4 個追加すれば，アクセラレータ全体が要するエネルギーをおおよそ一定に保つことができる．本評価では，積和演算ユニットを 32 個，専用計算ユニットを 0 個搭載した状態を初期状態として，積和演算ユニットが 0 個，専用計算ユニットが 128 個になるまで各ユニットを増減させる．

以上で述べたような方法によって各ユニット数を変更しながら推論処理を行う．CNN に 1 枚の画像を入力し出力を得るまでの時間は以下に示す式によって見積もる．

$$T = \sum_{l=1}^{\#layers} Max(DT_l, MT_l) \quad (1)$$

$$DT_l = Max(K_{l,1}, K_{l,2}, ..., K_{l,\#dedicated\_units}) \times Output_l \times Latency_{dedicated\_units} \quad (2)$$

$$MT_l = \left\{(K_{l,all} - \sum_{i=1}^{\#dedicated\_units} K_{l,i})/\#multiply\_units\right\} \times Output_l \times Latency_{multiply\_units} \quad (3)$$

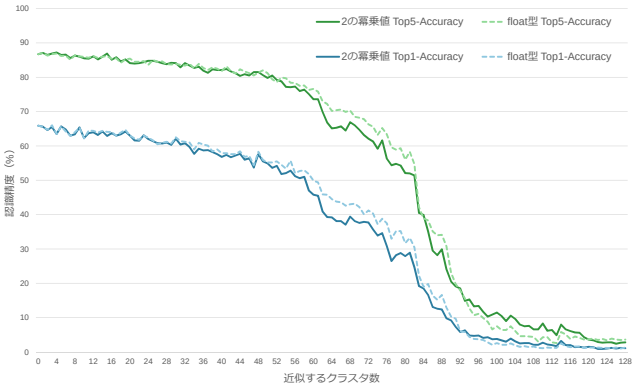

図 4　認識精度

(1) 式における T が CNN の出力計算に必要な時間を表しており，この T は CNN の各畳み込み層の出力計算に必要な時間の総和となる．各畳み込み層の出力は専用計算ユニットと積和演算ユニットで並列に計算することができ，それらの実行時間のうちの長い方が層全体の実行時間となる．そのため，層 l における専用計算ユニットの実行時間を DT_l，積和演算ユニットの実行時間を MT_l とし，それらのうちより大きい実行時間を全ての畳み込み層に対して累積する．専用計算ユニットを複数備える場合，DT_l は全ての専用計算ユニットの中で実行時間が最も大きいものの値となる．したがって，i 番目の専用計算ユニットに割り当てられたカーネル数 $K_{l,i}$ の中で最も大きい値に，出力マップサイズ $Output$ つまり畳み込み回数と，専用計算ユニットのレイテンシ $Latency_{dedicated\_units}$ を乗算することで求める．一方，MT_l は，全ての積和演算ユニットに，近似計算をしないカーネルを均等に割り当てて計算した際の時間となる．したがって，近似計算をしないカーネル数 $(K_{l,all} - \sum_{i=1}^{\#dedicated\_units} K_{l,i})$ を積和演算ユニット数 $Num_{multiply\_units}$ で除算した値に，出力マップサイズ $Output$ と積和演算ユニットのレイテンシ $Latency_{multiply\_units}$ を乗算することで求める．なお，2 の冪乗値への丸めによる近似を施す場合，カーネルの積和演算は単純な固定値シフト演算で実現できるため，専用計算ユニットのレイテンシ $Latency_{dedicated\_units}$ は 1 サイクルと仮定する．一方，積和演算ユニットの構成は様々考えられるため，今回は専用計算ユニットとのサイクル数の差が最低の場合を考え，積和演算ユニットのレイテンシ $Latency_{multiply\_units}$ は 2 サイクルと仮定する．

6.2 評価結果

6.2.1 認識精度

まず，**図 4** に認識精度の測定結果を示す．グラフの縦軸が認識精度，横軸が近似したクラスタ数を表している．評価結果は 4 本の折れ線グラフで表されており，緑色が Top5-Accuracy，青色が Top1-Accuracy を表す．また，点線がコモンカーネルの要素値を float 型のまま計算に使用

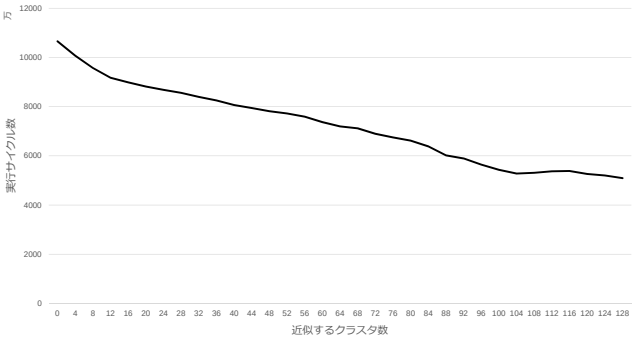

図 5 実行サイクル数

した場合，実線が 2 の冪乗値への丸めによる近似を行った場合の結果を表している．2 の冪乗値による近似を行った場合の結果に着目すると，20 クラスタあたりまで Top5-Accuracy および Top1-Accuracy の低下が 3%以内に抑えられていることが分かる．これは，既存の近似手法 [5] を用いて VGG-16 における推論処理を行った際に，再学習前の認識精度が数 10%低下していることを考慮すれば，実用的な値であると考えられる．なお，この 20 クラスタに含まれるカーネルの数は 778097 枚であり，CNN に含まれるカーネル全体の 47%にあたる．したがって，認識精度の低下を抑えつつ，多くのカーネルを近似計算することができると考えられる．また，2 の冪乗値の結果と float 型の結果を比較すると，50 クラスタあたりまでは認識精度にほとんど差がないことが分かる．このことから，今回の評価では 2 の冪乗値への丸めによる影響が小さいことが分かる．

6.2.2 計算時間

次に図 **5** に実行サイクル数の見積もり結果を示す．グラフの縦軸が実行サイクル数を万単位で表しており，横軸が近似したクラスタ数を表している．認識精度の評価で実用的な値が得られていた 20 クラスタの場合に着目すると，約 8800 万サイクルとなっている．専用計算ユニットを搭載しない場合は約 10600 万サイクルであるため，約 17%実行時間を削減できることが分かる．また，128 個全てのクラスタに含まれるカーネルを近似計算する場合には，実行時間を半分以下に削減可能であることが分かる．しかし，その場合は認識精度が実用的な値ではなくなってしまう．そのため，さらなるアクセラレータの性能向上には，認識精度を保つことができる近似方法の検討が必要である．

7. おわりに

本稿では，カーネルの類似性に基づいて近似計算を行う CNN アクセラレータを検討した．あらゆる CNN に多く含まれるカーネルを発見するために，カーネルクラスタリングを導入した．さらに，乗算器の代わりにコストの小さい演算器を使用可能にするために，カーネルの各要素を 2 の冪乗値へ丸めにことによって近似した．アクセーラレータの実現性を確認するために評価を行った結果，僅かな認識精度の低下で，計算時間を削減可能であることを確認した．しかし，近似計算を行うカーネルの数を増やすと，認識精度が大きく低下してしまった．したがって今後は，認識精度の低下を抑制することを目指し，クラスタリング方法および，近似対象とするするクラスタの選択方法を検討していく必要がある．

参考文献

[1] 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).

[2] Chen, Y. et al.: DaDianNao: A Machine-Learning Supercomputer, *Proc. 47th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-47)*, pp. 609–622 (2014).

[3] 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).

[4] Simonyan, K. and Zisserman, A.: Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556* (2014).

[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 preprint arXiv:1511.06530* (2015).

[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 preprint arXiv:1603.01025* (2016).

[8] Nakahara, H., Yonekawa, H., Sasao, T., Iwamoto, H. and Motomura, M.: A memory-based realization of a binarized deep convolutional neural network, *Field-Programmable Technology (FPT), 2016 International Conference on*, IEEE, pp. 277–280 (2016).

[9] Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y. and Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation, *Advances in Neural Information Processing Systems*, pp. 1269–1277 (2014).

[10] 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, *International Journal of Computer Vision (IJCV)*, Vol. 115, No. 3, pp. 211–252 (online), DOI: 10.1007/s11263-015-0816-y (2015).

[11] MacQueen, J. et al.: Some methods for classification and analysis of multivariate observations, *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1, No. 14, Oakland, CA, USA., pp. 281–297 (1967).

[12] Horowitz, M.: computing's energy problem (and what we can do about it), *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, IEEE, pp. 10–14 (2014).