Vol.2016-ARC-221 No.37
2016/8/10

再構成可能なニューラルネットワークアクセラレータの提案と性能分析

大場 百香[1,a)]　三輪 忍[1]　進藤 智司[2]　津邑 公暁[2]　八巻 隼人[1]　本多 弘樹[1]

概要：既存のニューラルネットワークアクセラレータは電力効率に優れているものの，特定のニューロンモデルや学習アルゴリズムしかサポートしていないため，人工知能研究者を始めとするさまざまなニューラルネットワークの計算を高速化したいユーザのニーズを満たせていない．そこで我々は，ニューラルネットワークアクセラレータに一般的に必要とされるカスタムロジックに加え，再構成可能ロジックを搭載したニューラルネットワークアクセラレータを開発している．このアクセラレータは，ユーザが再構成可能ロジックに所望の計算アルゴリズムを実装することによって，任意のニューラルネットワーク計算を可能にする．本論文では，上記のアクセラレータのアーキテクチャを提案するとともに，開発中のサイクル・レベルのシミュレータを用いて提案アクセラレータの性能分析を行った結果を報告する．

1. はじめに

画像認識や音声認識の分野で広く使われているニューラルネットワーク (NN) は，DeepLearning[11] の登場によって近年急速に大規模化が進み，その計算に要する時間や消費エネルギーが問題となっている．極端な例では，10 億個の結合重みからなる NN[11] やそれ以上の結合重みからなる NN[15] の計算が行われることもあり，これらの計算を実時間で行うためには 1,000 ノードを超える規模のコンピュータシステムを必要とする [11], [15]．この規模のシステムを用いたとしても計算には数日間を要することから [11]，大規模 NN の計算をより高速かつ低消費エネルギーで行うコンピュータシステムの実現が求められている．

このような背景から，現在，NN 計算のためのハードウェアアクセラレータ（NN アクセラレータ）が多数開発されている [1], [2], [3], [4], [5], [6], [7], [8], [10], [12], [13], [14], [16], [17], [18].

NN アクセラレータは，NN 計算のみが可能な専用ハードウェアである．オフチップ GPU と同様，ホストコンピュータに PCIe 等の手段で接続し，ホストコンピュータから計算をオフロードすることを想定して開発が進められている．NN アクセラレータは CPU や GPU と比べて 100 倍程度高速に NN 計算を行うことができる上，その消費エネルギーは CPU や GPU の 1/100 程度で済むことが知られている [13].

既存の NN アクセラレータは電力効率が優れている一方，アクセラレータとしてサポートするニューロンモデルや学習アルゴリズムが制限されており，その用途が限定的である．後述するように，NN を構成するニューロンのモデルには様々な種類があるが，ほとんどのアクセラレータはどれか 1 つのニューロンモデルの計算のみをサポートする [1], [2], [3], [4], [5], [6], [7], [8], [10], [12], [13], [14], [16], [17], [18]．また，大規模 NN では結合重みの学習に要する時間が膨大であるにも関わらず [11]，多くのアクセラレータは，与えられた入力パターンから出力パターンを計算する推論処理のみをサポートし，学習処理をサポートしていない [2], [5], [8], [14], [16], [17].

これは，応用目的で行われる NN 計算の多くは，同じニューロンモデルを使用しており，結合重みをオフラインで学習するためである [2], [4]．応用目的で NN 計算を行う場合には，アクセラレータのユーザがニューロンモデルの変更や結合重みの学習を行うことはほとんどない．

一方，人工知能研究者などの研究目的で NN 計算を行うユーザにとっては，既存アクセラレータの制約（ニューロンモデルを変更できない，学習処理をオフロードできない）は受け入れ難いものである．例えば，NN の学習アルゴリズムを開発する研究者にとっては，開発中の学習アルゴリズムを高速化することによってその有効性を早期に確認することが重要であり，この目的に対して学習処理をオフロードできないアクセラレータはほとんど役に立たない．あるいは，さまざまなニューロンモデルの優劣を短期間で

[1] 電気通信大学
[2] 名古屋工業大学
[a)] ohba@hpc.is.uec.ac.jp

比較評価したい研究者にとっては，1 つのニューロンモデルのみをサポートするアクセラレータは不便である．

そこで本研究では，ユーザによるニューロンモデルや学習アルゴリズムの変更が可能な NN アクセラレータ（再構成可能な NN アクセラレータ）を開発する．

我々が現在開発中の NN アクセラレータは，結合重みやニューロン値を格納するためのメモリ，ニューロンの出力値を計算するための積和演算器などの NN 計算に通常必要とされるカスタムロジックの他に，再構成可能ロジックを搭載することを想定している．各ユーザは，ニューロンモデル毎の出力計算アルゴリズムや学習アルゴリズムを再構成可能ロジック上に実装することによって，所望の NN 計算をアクセラレータ上で実行する．我々はこのような，任意の NN 計算を高速化かつ低消費エネルギー化するアクセラレータを開発する．

今回，再構成可能な NN アクセラレータのアーキテクチャの基本設計を行った．また，提案アクセラレータのサイクルレベルのシミュレータを現在開発しており，このシミュレータを用いて提案アクセラレータの性能分析を行った．その結果，Chen らの研究 [4] で行っている NN 計算と同じ出力計算を行った場合には，計算アルゴリズムをカスタムロジックの代わりに再構成可能ロジック上に実装することによる性能オーバヘッドはほとんどないことがわかった．本論文ではこれらの成果について報告する．

本論文の構成は以下のようになっている．まず 2 章でさまざまなアルゴリズムの NN が存在することを説明し，続く 3 章では既存の NN アクセラレータを紹介する．4 章では提案アクセラレータのアーキテクチャを説明する．5 章で提案アクセラレータの性能評価と分析を行い，6 章で本論文をまとめる．

2. ニューラルネットワーク

画像認識の分野では NN として CNN (Convolutional Neural Network) が有名であるが，CNN 以外にもさまざまな種類の NN[19] がこれまでに提案されている．本章ではニューロンモデル，ネットワークトポロジー，学習方法の観点から既存の NN についてまとめる．

2.1 ニューロンモデル

各ニューロンの出力は，一般的には以下のようにして計算される．まず，ニューロン i の膜電位 u_i は，ニューロン j の出力値 x_j とニューロン i,j 間の結合重み w_{ij} と閾値 θ_i を用いて，以下の式で求められる．

$$u_i = \sum x_j w_{ij} - \theta_i \tag{1}$$

上で求めた u_i に対し，後述する活性化関数 f を適用してニューロンの出力 y_i を得る．

$$y_i = f(u_i) \tag{2}$$

ニューロンモデルは以下の 2 種類に分類できる．

- 頻度モデル
  生体ニューロンでは，スパイク信号をニューロン間で伝達する．このスパイクの発生頻度を 0 ∼ 1 の数値で表現し，頻度情報がニューロン間で伝達されるとみなすのが頻度モデルである．したがって，頻度モデルでは式 (1) (2) の x_j, y_i は 0 ∼ 1 の値をとる．
  活性化関数としてはシグモイド関数がよく使われている．シグモイド関数 ς は，a を定数として以下の式によって定義される．

  $$\varsigma(x) = 1/(1 + e^{-ax}) \tag{3}$$

  ただし，シグモイド関数以外にも様々な活性化関数（ステップ，線形，双曲線，maxout，etc.）が提案されており，これらの活性化関数が使用されることもある．
  ニューロンの出力計算において，式 (1) の積和演算を行わない場合もある．その場合，ニューロン i の入力値の集合 $x_0, x_1, \cdots$ から出力 y_i を別の方法によって計算する．例えば，入力値 $x_0, x_1, \cdots$ の中から最大値を選ぶ，入力値 $x_0, x_1, \cdots$ の平均を計算して出力する，などの方法がある．
- スパイキングニューロンモデル
  頻度モデルよりも生体ニューロンの振る舞いに近いモデルとして，スパイクの発生過程をモデル化したスパイキングニューロンモデルがある．スパイキングニューロンモデルのニューロン間で伝達される信号はスパイクであり，スパイクの有無を 1 または 0 の信号によって表現する．
  膜電位はさまざまな計算方法が提案されており，例えば，以下の式によって計算する．

  $$V_j(t) = V_j(t-1) - \lambda_j + \sum w_{i,j} \times x_i(t) \tag{4}$$

  上の式において，$V_j(t)$ は時刻 t におけるニューロン j の膜電位の値，λ_j はニューロン j の膜電位の減衰定数，$x_i(t)$ は時刻 t においてニューロン i で発生したスパイクの有無を表している．膜電位 $V_j(t)$ が閾値を超えるとニューロン j はスパイクを発生し，その時の出力 $y_j(t)$ は 1 となる．なお，スパイクが発生しない時は，ニューロンの出力は 0 となる．

2.2 ネットワークの種類

- フィードフォワード型
  ループが存在しない NN を Feedforward 型の NN という．**図 1** 左に示すようにニューロンは層状に並んでおり，入力層 → 中間層 → 出力層のように単一方向へのみ信号が伝播する．
- フィードバック型
  ループが存在する NN を Feedback 型の NN という．

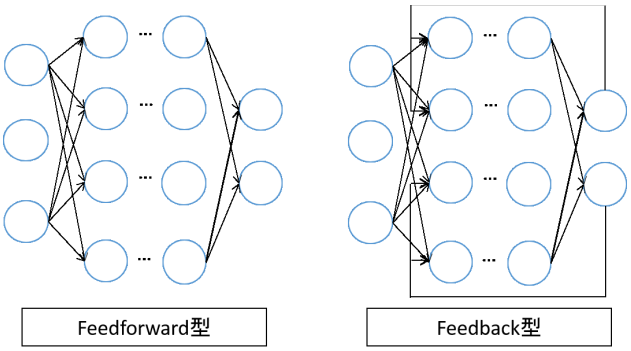

図 1 ネットワークの種類

図 1 右では，出力層のニューロンから 2 層目のニューロンへと信号が戻るループがある．中には，各ニューロンが他のすべてのニューロンと結合した NN もある．

2.3 学習アルゴリズムの種類

ここでは学習アルゴリズムの種類をまとめる．

- 教師あり学習
  教師あり学習では，学習用に入力データと教師データのセットを用意し，入力データに対する NN の応答が教師データに近づくように結合重みを修正する．NN の学習アルゴリズムとして有名な誤差逆伝播法は，教師あり学習である．誤差逆伝播法では，出力層から入力層に向かって誤差信号を伝播させていくことによって，各結合重みの変化量を計算する．その際に積和演算が行われる．
- 教師なし学習
  教師なし学習では，入力データのみを用いて，ある学習則にしたがって結合重みを修正する．結合重みの修正方法としては，例えば，隣接する 2 つのニューロンがともに短期間にスパイクを出力した場合は結合重みを増やす，などの方法がある．ローカルな信号変化に基づいて結合重みを修正することで，入力データに含まれる何らかのパターンや特徴，法則を NN が自律的に発見することを期待するのが教師なし学習である．

3. 既存のニューラルネットワークアクセラレータ

本論文の冒頭で述べたように，既存の NN アクセラレータは積和演算，活性化関数などのニューロンの出力計算をロジック化しており，異なるニューロンモデル，学習アルゴリズムには対応できないことが特徴的である．

一方，既存アクセラレータでは電力効率を向上させるためのさまざまな工夫が行われており，例えば，メモリを演算ユニットの近くに配置してメモリ・アクセスに必要なレイテンシや消費エネルギーを抑制する，結合重みではなくニューロン値を移動させることで消費エネルギーを抑制する，などが行われている．本章では既存アクセラレータを紹介する．

3.1 DianNao

DianNao[2] はメモリ転送の回数，消費エネルギーを減らすことを目的とし設計された，大規模 NN に適応できるアクセラレータである．

DianNao は，結合重みとニューロン値を保持する外部メモリ，結合重みとニューロン値を一時的に保持するためのバッファ，大量の積和演算回路と活性化関数の計算回路からなる NFU，制御プロセッサで構成されている．制御プロセッサは VLIW のような 512 ビットの専用命令を解釈・実行する．上記の専用命令 1 命令が制御プロセッサで実行されると，外部メモリから結合重みと入力ニューロンの値（どちらもベクトル・データ）がバッファにロードされ，NFU を用いてニューロンの出力計算を並列に行い，計算結果をバッファまたは外部メモリに書き込む，という一連の処理が行われる．なお，DianNao が実行できるのは NN の出力計算，すなわち，推論処理のみであり，学習処理は実行できない．

その他の特徴として，DianNao は，バッファごとに DMA(Direct Memory Access) エンジンを設けることで，メモリとバッファ間のデータ転送とニューロンの出力計算を並列に実行できるようにしており，これによって高スループットを実現している．また，バッファ上のニューロン値を再利用することでメモリ・アクセスを減らし，省エネルギー化を実現している．

3.2 DaDianNao

DaDianNao[4] は，DianNao の問題点である計算ユニット（NFU）とメインメモリ間のバンド幅の不足を改善し，大規模 NN 計算の更なる高速化と省エネルギー化を目的として設計されたアクセラレータである．

DaDianNao は複数のコアによって構成されており，各コア（**図 2**）が複数ニューロンの出力計算を並列に行う．ニューロンの出力計算を行うユニット (NFU) はカスタムロジックによって構成されている．結合重みは各コアが備える結合重みバッファに格納されている．

一方，ニューロンの入力値は図 2 の左側に位置する共有メモリに格納されており，必要なデータが制御プロセッサによって入力バッファにロードされる．NFU は結合重みバッファと入力バッファの値（どちらもベクトル・データ）を用いて複数ニューロン（16 個程度）の出力計算を並列に行い，計算結果（ベクトル・データ）を出力バッファへ出力する．

なお，図 2 に示すように，NFU 内は 3 つのパイプライン・ステージに分かれており，出力計算はパイプライン処理される．このように，DaDianNao のコア内の動作は，共有メモリからの結合重みのロードが不要な点を除いて

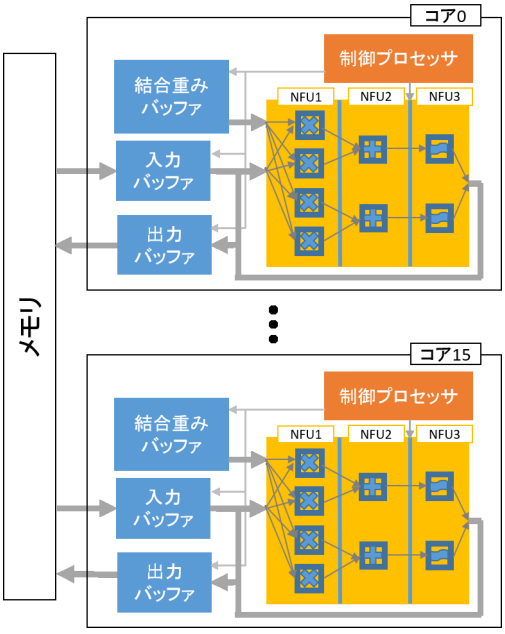

図 2 DaDianNao のアーキテクチャ

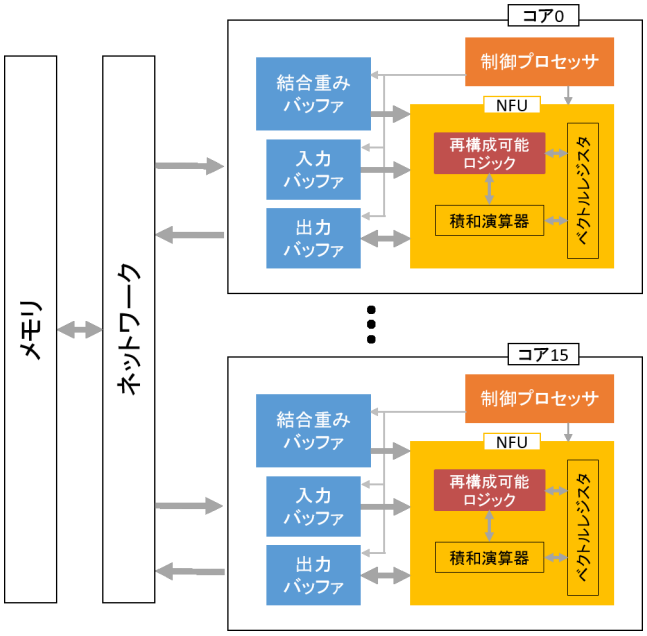

図 3 提案アーキテクチャ

DianNao とほぼ同じである．

DaDianNao は，このように計算に必要なデータを格納するメモリを NFU の近くに配置することで，これらのデータのアクセスレイテンシとアクセスエネルギーを抑制している．DaDianNao は推論だけでなく学習も可能である．

3.3 TrueNorth

TrueNorth[14] は，スパイキングニューラルネットワークをハードウェア化したアクセラレータである．

TrueNorth の各コアは，結合重みと膜電位の値を保持するメモリ，膜電位とスパイクの計算を行うニューロン，制御用のコントローラ，コア間でスパイクを転送するルータ，スパイクを保持するスケジューラで構成されている．

前述の DianNao と DaDianNao は命令駆動であったが，TrueNorth はイベント駆動で動作する．コントローラがスケジューラをチェックし，スパイクが到着していたらニューロンをメモリから順に読みだして処理を行う．推論のみ実行できる．

4. 再構成可能なニューラルネットワークアクセラレータ

4.1 提案アーキテクチャ

我々は，任意の NN 計算をサポートするため，再構成可能ロジックを搭載した NN アクセラレータを提案する．提案アクセラレータのアーキテクチャを**図 3** に示す．

基本的なアーキテクチャは DaDianNao（図 2）と同様であり，複数コアがネットワークを介して接続されており，1 つのメモリを共有した構成となっている．各コアは，NFU，制御プロセッサ，結合重み，入力ニューロン，出力ニューロンそれぞれの値を保持するバッファによって構成される．提案アクセラレータが DaDianNao と異なるのは，再構成可能ロジックと，演算途中の結果を一時的に格納するためのベクトル・レジスタが NFU 内に存在する点である．

再構成可能ロジックは，積和演算以外の NN 計算（例えば，活性化関数の計算，膜電位の減少量とスパイク発生の計算など）をハードウェアで実現する．積和演算は多くの NN 計算で必要とされるため，カスタムロジックとして構成する．

ベクトル・レジスタはニューロンの状態や膜電位を保持する．これら NFU 内のロジックはすべてパイプライン化されているものとする．

制御プロセッサは，他のコアへのデータ転送，推論と学習のどちらを行うかコアに対して指示する．DaDianNao と同じような構成にすることで，共有メモリと NFU 間の結合重みの転送を省略し，転送に必要なレイテンシと消費エネルギーを抑制する．

提案アーキテクチャの制御プロセッサ内部の構造は，DaDianNao とは異なる．提案アーキテクチャの制御プロセッサも，共有メモリからニューロン値を入力バッファへロードし，NFU で計算を行い，結果を出力バッファまたは共有メモリに書き込む，という一連の処理を行う長命令を実行する点は DaDianNao と同じである．それに加え，提案アーキテクチャの制御プロセッサは，一部のスカラ命令（分岐命令や整数系の四則演算命令）も実行できる．このようにすることで，ループ構造を持った制御プログラムの記述を可能にし，プログラム中の静的命令数を削減できる．

図 2 には示されていないが，制御プロセッサ内にはプログラム格納用のメモリが存在しており，このメモリのサイ

ズを小さくできると期待している．上記のスカラ命令を実行するために，これも図 2 には示されていないが，制御プロセッサ内には少量のスカラ・レジスタが存在する．

4.2 ニューラルネットワークアクセラレータシミュレータ（NNA-Sim）

提案アーキテクチャのサイクル・レベルのシミュレーションを行うために，我々は NNA-Sim を開発している．NNA-Sim は，結合重み，プログラム，ニューロン値の初期データが，それぞれ，重みバッファ，プログラム格納用メモリ，共有メモリにセットされた状態からシミュレーションを開始する．初期データをホストコンピュータからアクセラレータに転送し，アクセラレータ内の各メモリにセットする処理は実行サイクル数としてカウントしない．NNA-Sim はまだ開発途中であり，現在は推論のみ実行可能で学習はできない．

5. 評価

5.1 評価内容と評価方法

提案アクセラレータと DaDianNao は，NFU 以外の大部分の構成が同じであるため，シミュレーション条件を変更すれば NNA-Sim で DaDianNao の性能シミュレーションを行うことも可能である．そこで，まず NNA-Sim のシミュレーション精度を検証するために，DaDianNao の論文で行われている性能評価実験の再現実験を行った．この再現実験によって NNA-Sim が十分な精度を有していることを確認できたため，続いて，提案アクセラレータの性能分析を NNA-Sim を用いて行った．

評価対象とする NN は，DaDianNao の論文で使用されていた評価用 NN の一部を使用する．DaDianNao の論文では計 10 種類の NN を評価していたが，今回は時間の都合により，**表 1** に示した 6 種類の NN のみを評価した．残りの NN の評価は今後の課題である．表 1 の N_x，N_y は 3 次元構造の x 方向と y 方向のニューロン数を表し，K_x，K_y はカーネルの大きさを表す．N_z^i，N_z^o は 3 次元構造の z 方向のニューロン数を表す．

今回評価に使用した NN は，画像認識等で実際に使用されている CNN の 1 つの層を取り出したものであり，Convolutional 層（CONV），Pooling 層（POOL），Classifier 層（CLASS）の 3 種類からなる．

表 1 評価対象の NN

layer	N_x	N_y	K_x	K_y	N_z^i	N_z^o
CLASS1	-	-	-	-	2560	2560
CLASS2	-	-	-	-	4096	4096
CONV1	256	256	11	11	256	256
CONV2	500	375	9	9	32	48
POOL1	492	367	2	2	12	12
POOL2	256	256	2	2	256	256

CONV 層は，入力データの特徴要素を識別する層である．入力ニューロン（$N_x \times N_y \times N_z^i$）に対して 2 次元のカーネル（$K_x \times K_y$）を 1 ニューロンずつスライドさせて出力計算を行う．そのため，出力ニューロンは（$(N_x - K_x + 1) \times (N_y - K_y + 1) \times N_z^o$）の 3 次元構造となる．

POOL 層は CONV 層の直後に配置される層であり，入力されたデータ（$N_x \times N_y \times N_z^i$）にカーネル（$K_x \times K_y$）を K_x または K_y ニューロンずつスライドさせながら出力を計算する．そのため，出力ニューロンは $(N_x/K_x) \times (N_y/K_y) \times N_z^o$ の 3 次元構造となる．

CLASS 層は CONV 層と POOL 層をまとめる層であり，1 次元に配置された入力ニューロン（N_z^i）と，同じく 1 次元に配置された出力ニューロン（N_z^o）が全結合されている．

再現実験では，先行研究と同様，CPU，GPU，アクセラレータそれぞれにおいて上記の NN の推論処理に要する時間を測定し，比較した．CPU は Intel Xeon E5-2630 v2 を，GPU は NVIDIA Tesla K20M を使用した．CPU と GPU における推論時間の測定には，ディープラーニングのフレームワークである Caffe[9] を使用した．アクセラレータにおける推論時間は NNA-Sim を用いて求めた．NNA-Sim ではコア数，共有メモリ・サイズ等のアーキテクチャ・パラメータを設定することができる．これらの値は DaDianNao の論文を参考に**表 2** の値とした．また，NFU 全体のレイテンシは 2 サイクル（積和演算に 1 サイクル，活性化関数の計算に 1 サイクル）とした．

DaDianNao と同様，NNA-Sim では，共有メモリへのアクセスが DMA によってパイプライン処理されることを想定している．今回の評価では，入力バッファの DMA が処理可能なメモリリクエスト数の上限をコアあたり 64 とした．メモリリクエスト数がこの上限を超えると，DMA は新たなメモリリクエストを受けつけることができなくなり，ロードを行う命令がデコードされた時点で制御プロセッサがストール（LoadBlocking）する．なお，今回の評価では，出力バッファの DMA には処理可能なメモリリクエストの上限値を設けていない．これは，出力バッファの DMA が処理する書き込みリクエスト数は，入力バッファの DMA が処理する読み出しリクエスト数よりも少ないためである．

表 2 シミュレーション・パラメータ

パラメータ	設定値
コア数	16
共有メモリ	16 ポート，4MB，10 サイクル
入力バッファ	16 エントリ，1 サイクル
出力バッファ	16 エントリ，1 サイクル
重みバッファ	2M エントリ
オンチップ・ネットワーク	100GB/s，10 サイクル
周波数	606MHz

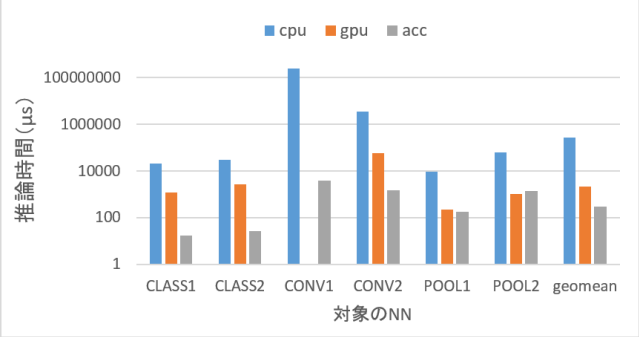

図 4 性能比較結果

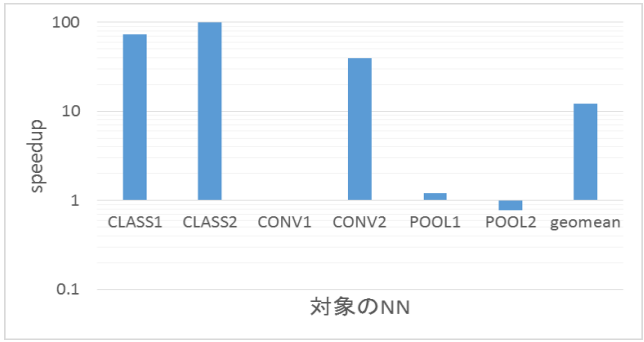

図 5 アクセラレータの高速化率（対 GPU）

前章で述べたように，提案アクセラレータは，NFU 内に再構成可能ロジックを有し，この再構成可能ロジック上に NN 計算のアルゴリズムをユーザが実装することを想定している．そのため，DaDianNao を始めとする，NN 計算がカスタムロジックで構成されたアクセラレータと比べ，計算時間が長くなると予想される．

そこで，今回，提案アクセラレータにおいて積和演算回路が再構成可能ロジック上に実装されたことを想定し，積和演算回路のレイテンシを 1, 10, 100 サイクルと変化させた場合の性能評価と分析を行った．また，後述するように，提案アクセラレータの性能は，オンチップ・ネットワークのバンド幅に強く依存する．そこで，バンド幅を 100, 150, 200GB/s と変化させた場合の性能評価と分析も行った．

5.2 再現実験

図 4 に比較評価の結果のグラフを示す．グラフの横軸は評価対象の NN，縦軸は実行時間（単位はマイクロ秒）を表している．NN ごとの 3 本の棒グラフは，左から順に，CPU，GPU，アクセラレータ（DaDianNao）の実行時間を表している．なお，図には CONV1 を GPU で実行した時の実行時間が表示されていないが，これは，メモリ容量不足により GPU 上で CONV1 を実行できなかったためである．グラフより，GPU は CPU に対して平均 124 倍，アクセラレータは GPU に対して平均 7 倍高速であることがわかる．

DaDianNao の論文の結果と比較するため，図 4 から GPU とアクセラレータの実行時間のみを抽出し，GPU に対するアクセラレータの高速化率をグラフにしたものが **図 5** である．文献 [4] の図 10 と比較すると，NNA-Sim を用いて評価した DaDianNao の性能は，POOL2 以外は先行研究とほぼ同程度の性能を示しており，我々が開発した NNA-Sim は DaDianNao のサイクル・レベルのシミュレータとして十分な精度を有していることがわかる．

なお，POOL2 において，我々の実験結果が文献 [4] の図 10 と一致しないのは，GPU の実行時間の測定に使用したプログラムが Chen らの研究 [4] とは違うことが原因と考えている．先行研究では測定に Caffe ではなく自作のプログラムを使用しているが，このプログラムが Caffe と同程度のチューニング施されておらず，その結果として，GPU に対する高速化率に違いが出たのではないかと推測している．

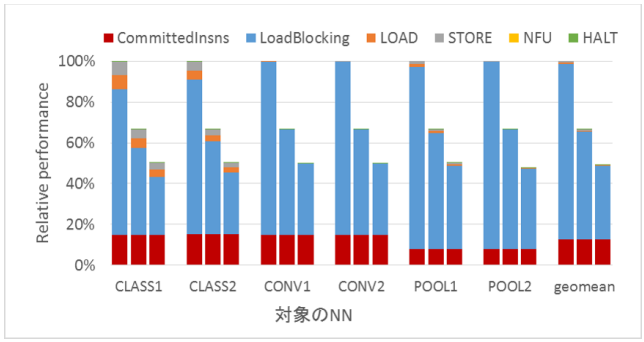

図 6 オンチップ・ネットワークのバンド幅を変更した場合の実行サイクル数の内訳（積和演算回路のレイテンシ：1）

5.3 性能分析

オンチップ・ネットワークのバンド幅を変化させた場合の実行サイクル数の内訳を**図 6** に示す．グラフの横軸は NN を，縦軸は実行サイクル数を表している．NN ごとの 3 本の積み上げ棒グラフは，左から順に，バンド幅が 100, 150, 200GB/s の時の結果である．各積み上げ棒グラフは 6 つの項目からなり，下から順に，総実行命令数（CommittedInsns），ロードがストールしたサイクル数（LoadBlocking），ロードの実効サイクル数（LOAD），ストアの実効サイクル数（STORE），NFU の実効サイクル数（NFU），プログラムが終了して停止していた（他コアの計算が終了するのを待っていた）サイクル数（HALT）を表している．なお，各サイクル数は，16 コアの平均値であり，100GB/s の時の総実行サイクル数で正規化してある．

図 6 より，LoadBlocking が総実行サイクル数の大半を占めていることがわかる．LoadBlocking は，100GB/s のオンチップ・ネットワークを仮定した場合，総実行サイクル数の平均 86%を占める結果となった．LoadBlocking は，ネットワーク・バンド幅の増加とともに減少するが，200GB/s のオンチップ・ネットワークを仮定した場合でも，総実行サイクル数の平均 73%（100GB/s の時の総実行

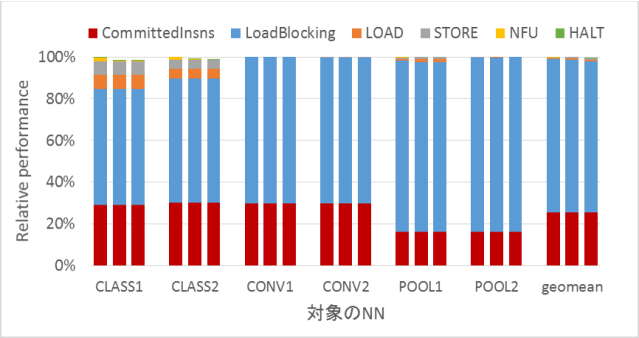

図 7 積和演算回路のレイテンシを変更した場合の実行サイクル数の内訳（オンチップ・ネットワークのバンド幅：200GB/s）

サイクル数で正規化していることから，200GB/s の総実行サイクル数は 49%，LoadBlocking は 36%となる）を占めている．以上のことから，提案アクセラレータでは共有メモリに対しリクエストを待つ時間が性能上のボトルネックとなっており，この点を今後解決していかなければならないことがわかる．

図 7 は，ネットワーク・バンド幅を 200GB/s として，積和演算回路のレイテンシを変更した場合の実行サイクル数の内訳である．NN ごとの 3 本の積み上げ棒グラフは，左から順に，積和演算回路のレイテンシが 100, 10, 1 サイクルの時の結果を表している．図 6 と同様，各実行サイクル数は， NN ごとに左端（100 サイクルの時）の総実行サイクル数で正規化されている．

グラフより，積和演算回路のレイテンシが増加しても，総実行サイクル数はほとんど変わらない．積和演算回路のレイテンシが 1 サイクルの場合と 100 サイクルの場合との総実行サイクル数の差は，平均 1%，最大 2%（CLASS1）であった．総実行サイクル数に差が見られない理由は，提案アクセラレータでは NFU がパイプライン化されることを想定しているためである．今回評価した NN はパイプライン・ハザードを生じさせないものであったため，積和演算回路のレイテンシの増加が総実行サイクル数に影響を与えなかったと考えられる．パイプライン・ハザードが生じる NN の評価は今後の課題である．

6. おわりに

本論文では, 任意の NN に対応できる再構成可能なニューラルネットワークアクセラレータの提案とその評価, 性能分析を行った．提案アクセラレータでは共有メモリに対するリクエストを待つ時間が性能上のボトルネックとなっていることから，今後はボトルネックを解消するための手法を検討する．また，パイプライン・ハザードが生じる NN の評価も行う予定である．

謝辞 本研究の一部は JST CREST による．

参考文献

[1] Albericio, J., Judd, P., Hetherington, T., Aamodt, T., Jerger, N. E. and Moshovos, A.: Cnvlutin: Ineffectual-Neuron-Free Deep Convolutional Neural Network Computing, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 1–13 (2016).

[2] Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y. and Temam, O.: DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning, *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pp. 269–284 (2014).

[3] Chen, Y.-H., Emer, J. and Sze, V.: Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 367–379 (2016).

[4] Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N. and Temam, O.: DaDianNao: A Machine-Learning Supercomputer, *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, pp. 609–622 (2014).

[5] Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y. and Xie, Y.: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 27–39 (2016).

[6] Du, Z., Ben-Dayan Rubin, D. D., Chen, Y., He, L., Chen, T., Zhang, L., Wu, C. and Temam, O.: Neuromorphic Accelerators: A Comparison Between Neuroscience and Machine-learning Approaches, *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, pp. 494–507 (2015).

[7] Du, Z., Fasthuber, R., Chen, T., Ienne, P., Li, L., Luo, T., Feng, X., Chen, Y. and Temam, O.: ShiDianNao: Shifting Vision Processing Closer to the Sensor, *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, ISCA '15, pp. 92–104 (2015).

[8] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. and Dally, B.: EIE: Efficient Inference Engine on Compressed Deep Neural Networkg, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 243–254 (2016).

[9] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding, *arXiv preprint arXiv:1408.5093* (2014).

[10] Kim, D., Kung, J., Chai, S., Yalamanchili, S. and Mukhopadhyay, S.: Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 380–392 (2016).

[11] Le, Q. V., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J. and Ng, A. Y.: Building high-level features using large scale unsupervised learning, *In International Conference on Machine Learning* (2012).

[12] LiKamWa, R., Hou, Y., Polansky, M., Gao, Y. and Zhong, L.: RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision, *Proceedings of the 43rd Annual International Symposium on Com-*

puter Architecture, ISCA '16, pp. 255–266 (2016).

[13] Liu, S., Du, Z., Tao, J., Han, D., Luo, T., Xie, Y., Chen, Y. and Chen, T.: Cambricon: An Instruction Set Architecture for Neural Networks, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 393–405 (2016).

[14] Merolla, P., Arthur, J., Alvarez-Icaza, R., Cassidy, A., Sawada, J., Akopyan, F., Jackson, B., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S., Appuswamy, R., Taba, B., Amir, A., Flickner, M., Risk, W., Manohar, R. and Modha, D.: A million spiking-neuron integrated circuit with a scalable communication network and interface, *Science*, pp. 668–673 (2014).

[15] Preissl, R., Wong, T. M., Datta, P., Flickner, M., Singh, R., Esser, S. K., Risk, W. P., Simon, H. D. and Modha, D. S.: Compass: A Scalable Simulator for an Architecture for Cognitive Computing, *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pp. 54:1–54:11 (2012).

[16] Reagen, B., Whatmough, P., Adolf, R., Rama, S., Lee, H., Lee, S., Lobato, J. M. H., Wei, G.-Y. and Brooks, D.: Minerva: Enabling Low-Power, High-Accuracy Deep Neural Network Accelerators, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 267–278 (2016).

[17] Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S. and Srikumar, V.: ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars, *Proceedings of the 43rd Annual International Symposium on Computer Architecture*, ISCA '16, pp. 14–26 (2016).

[18] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B. and Cong, J.: Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks, *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, pp. 161–170 (2015).

[19] 熊沢逸夫：学習とニューラルネットワーク（電子情報通信工学シリーズ），森北出版 (1998).