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.

深層畳込みニューラルネットワークに向けた データ流再構成型演算器アレイアーキテクチャ

安藤　洸太† 　折茂健太郎† 　植吉　晃大† 　浅井　哲也† 　本村　真人†

† 北海道大学 大学院情報科学研究科　〒 060-0814 北海道札幌市北区北 14 条西 9 丁目
E-mail: †ando@lalsie.ist.hokudai.ac.jp

あらまし　近年、畳込みニューラルネットワーク (CNN) による大規模な機械学習が急速な発展を見せ、主に画像認識の分野で成果を挙げている。CNN は非常に演算量が大きく、旧来の CPU による逐次処理では膨大な時間を要するため、種々のハードウェアアクセラレータが提案されている。しかしそれらは CNN の二大要素である畳込み層と全結合層の処理内容の差異に起因して汎用性に問題がある。そこで本研究では単純な演算素子を多数並列して広帯域のバスで結び、各処理に応じて制御することで様々な CNN の畳込み層と全結合層の処理に対応できるアクセラレータを提案する。

キーワード　CNN, 畳込みニューラルネットワーク, ニューラルネットワーク, 全結合, アクセラレータ, 並列処理

1. はじめに

近年、畳込みニューラルネットワーク (CNN; Convolutional Neural Network) による機械学習が急速な発展を見せ、主に画像認識の分野で成果を挙げている。CNN は多層ニューラルネットワークの一部に 2 次元畳込み演算を取り入れ、特徴抽出を内包したものといえる。人手を介さず特徴抽出をも学習することで高い汎化性能を実現している。

CNN は旧来のニューラルネットワークに比べ演算量が大きく、CPU による逐次処理では膨大な時間を要するため、様々なアクセラレーション手法が提案されてきた。GPU による並列演算 (GPGPU 処理) が実用化され [1]、ソフトウェアベースの高い汎用性と高速な数値演算が CNN の学習を支えているが、電力効率の向上が難しい。またターゲットを絞って専用アクセラレータ LSI(ASIC) を作製する研究 [2] も盛んであり、専用回路による無駄のない高速・高効率性が利点であるが、対応可能な応用が固定されて進展の早いディープラーニングの研究に追随し難いのが難点である。各応用に最適化した回路を FPGA 上に実装する研究 [3] も存在し、処理効率と汎用性の両立を目指している。

先行研究の一例として、MIT による Eyeriss アーキテクチャ [4] は CNN を主たる応用先として設計されたアクセラレータ LSI で、データの再利用を考慮した並列演算器で畳込み処理を行う。巧妙な内部データ配分と並列プロセッサにより畳込み処理を 300 mW 程度で処理することができるが、全結合層の処理においては演算対転送比が大きく低下してしまう。また UCLA の提案 [3] は演算とデータ転送の比率に着目し、使用アプリケーションに最適な並列パラメータを探って高位合成による FPGA 実装を試みている。

しかしこれまでの手法の難点となっているのは、CNN の二大要素である畳込み層と全結合層の処理内容の差異に起因する汎用性の低下である。CNN の処理時間の内では畳込み層の占める割合が大きいが、畳込み層だけに注目して全結合層の処理能力を犠牲にするのは結果的にシステム全体の性能低下の原因となりうる。本研究では CNN の汎化性能を支え演算量の大半を占める畳込み層と、分類処理で欠かすことができない全結合層の両方において性能を発揮するハードウェアアーキテクチャを検討する。単純な演算素子を多数並列して広帯域のバスで結び、それを畳込みと全結合との処理段階に応じて制御することで、様々な CNN の畳込み層と全結合層の処理に柔軟に対応できるアクセラレータを提案する。

2. 畳込みニューラルネットワークのデータ流の解析

CNN は主に二次元構造を持った入力 (画像等) に対し二次元の係数を空間畳込みする処理を含む多層ニューラルネットワークである。画像に対する二次元畳込みは、係数 (フィルタカーネル) のパターンに類似のパターンが現れる画素を強調する特徴抽出を行うものといえる。CNN 内部の畳込み演算は学習によって目的のタスクに最適な係数に調整され、特徴抽出器として働く。従来人力で調整されていた特徴抽出部分を含めて学習を行うことができることが CNN がコンピュータビジョン等の自然データの処理で成果を挙げている一因である。

CNN の多層構造の中で、二次元畳込みに相当する処理を行う層を畳込み層といい、旧来のニューラルネットワークと同様に一次元の行列変換に相当する処理となる層を全結合層という。

性能や面積を犠牲にせず畳込みと全結合の処理を統一的に扱う共用アーキテクチャを考案するに先立ち、それぞれのデータ

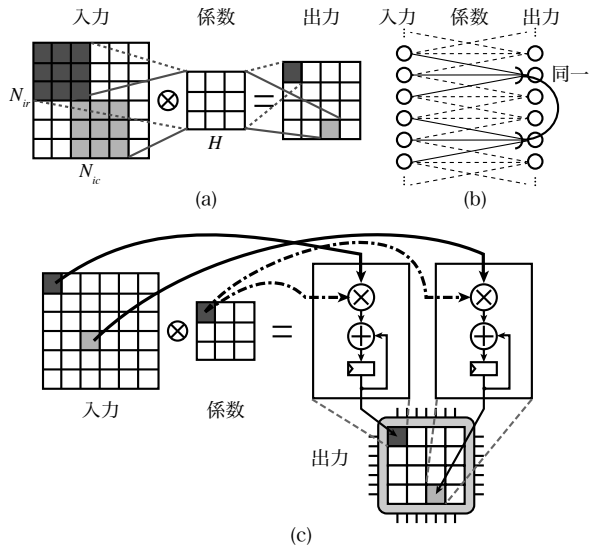

図 1　畳込み層のデータ構造 (a)(b) と並列化 (c)

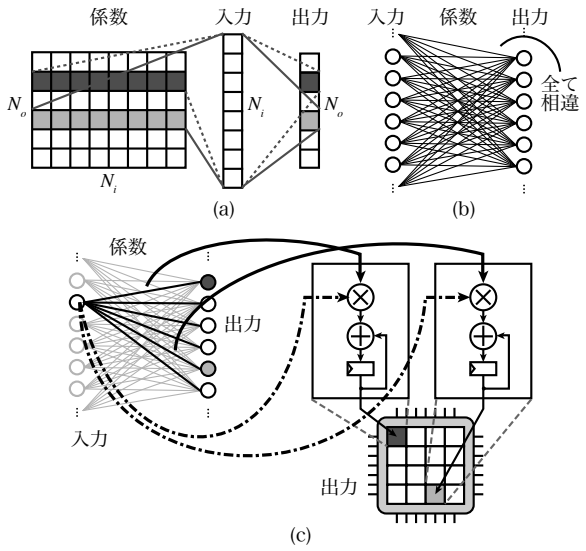

図 2　全結合層のデータ構造 (a)(b) と並列化 (c)

構造・データパスを解析し、根本的な特徴を捉える。

2.1　畳込み層のデータ構造

畳込み層のデータ構造を図 1(a)(b) に示し、一般的な演算内容を説明する。図 1(a) のように、サイズ $H \times H$ の二次元の係数を、同じく二次元で $N_{ir} \times N_{ic}$ の入力平面に重ねて $H \times H$ の部分領域を取る (図 1(a) では $H \times H = 3 \times 3$, $N_{ir} \times N_{ic} = 6 \times 6$)。この部分領域と係数の対応する要素同士の積の総和が出力平面の新しい 1 要素の値となる。このとき係数はストライド s (注1) で入力平面をスライドし、入力平面全体をなめるように部分領域をとるが、各部分領域の間で係数は共通である。これらの二次元のデータ構造を崩して一次元のニューラルネットの形に書き下すと、図 1(b) のように各出力ニューロンがある特定の位置関係にある入力ニューロンのみと結合し、それぞれの重み係数は共通する構造となる。

このように畳込み層の演算は同数の乗算と加算で成り立っている。ここで各出力要素を決定するために必要なデータを考える。出力平面の全要素は同じ係数 (群) を共有しているので、全出力要素の計算は必ず各係数要素との乗算を含んでいる。一方各出力要素が関係を持つ入力平面の領域は図 1(a) で見たように出力要素の位置と係数平面の大きさ (H) で決まる。係数を一つ選んだとき、各出力要素それぞれへの入力要素はそれぞれ一つに定まる。よって、全出力に共通の係数を一つ選び、出力要素群が関係している入力要素群を抜き出して乗算する操作を、全係数に対して順に行って加算することで畳込みの演算が完了する。この一度の乗算と加算で、全出力要素に対しただ一つの係数と、出力要素と同数の入力要素が要求される (図 1(c))。

2.2　全結合層のデータ構造

これと対比して全結合層の構造を見てみる (図 2(a)(b))。図 2(b) のように、一つの出力要素には全ての入力要素からの重み付き総和が入力されるので全結合型と呼ばれる。この全ての入力要素から全ての出力要素への結合の計算はベクトルと行列の積の形で表すことができる。図 2(a) のように、1 出力要素の値は係数行列の 1 行と入力の要素積の和 (内積) である。

ここでも基本となる演算は乗算と加算である。この演算に必要なデータ供給を考える。先に述べた通り 1 出力要素には全入力要素がそれぞれの係数で接続されている。逆に言えば 1 入力要素はそれぞれの重みで全出力要素に影響を持っている。よって、一度に一つの入力要素を取り出し、その入力から全出力要素への係数群 (係数行列の 1 列) を切り出す処理を、全入力要素に対して繰り返すことで、全結合層の内積計算を行うことができる。このときは全出力要素について一度の乗算と加算で、ただ一つの入力要素と、出力要素と同数の係数要素が必要となっている (図 2(c))。

3.　畳込み層・全結合層 共用アーキテクチャの提案

前章の解析結果から、畳込み層と全結合層の演算自体は同じ積和計算であるものの、出力要素 (群) を基準としてそれぞれの演算に必要なデータを考えたとき、畳込み層では多数の入力要素と一つの係数要素、全結合層では一つの入力要素と多数の係数要素と逆の関係となっていることに気づく。これを利用して、畳込み層と全結合層で大幅な構成の差異を要することなく単一のハードウェア構成で処理を行うアーキテクチャを構築する。

3.1　基本アーキテクチャ

畳込み層と全結合層はどちらも積和算が基本であるので、乗算アキュムレータを基本とした小規模で単純な積和演算器でアレイを構成する。1 演算器は 1 出力要素を扱う出力要素並列である。ここで前章の検討の結果から、演算の度に「一つの共通データと多数の別々のデータ」を配分する必要がある。全演算器 (=出力要素) に共通のデータを入力する共通入力と各演算器に別々のデータを与える個別入力の 2 系統の入力を持たせることで、畳込み層と全結合層の両方に対応できるだろう。

畳込み層の処理 (図 3(a)) では各演算器に個別の入力要素を与え、それとは別に全演算器共通の係数を配分する。出力平面

(注1)：係数を入力平面上でスライドさせる間隔、言い換えれば一つの係数要素が使う複数の入力要素間の間隔をストライドと呼ぶ。

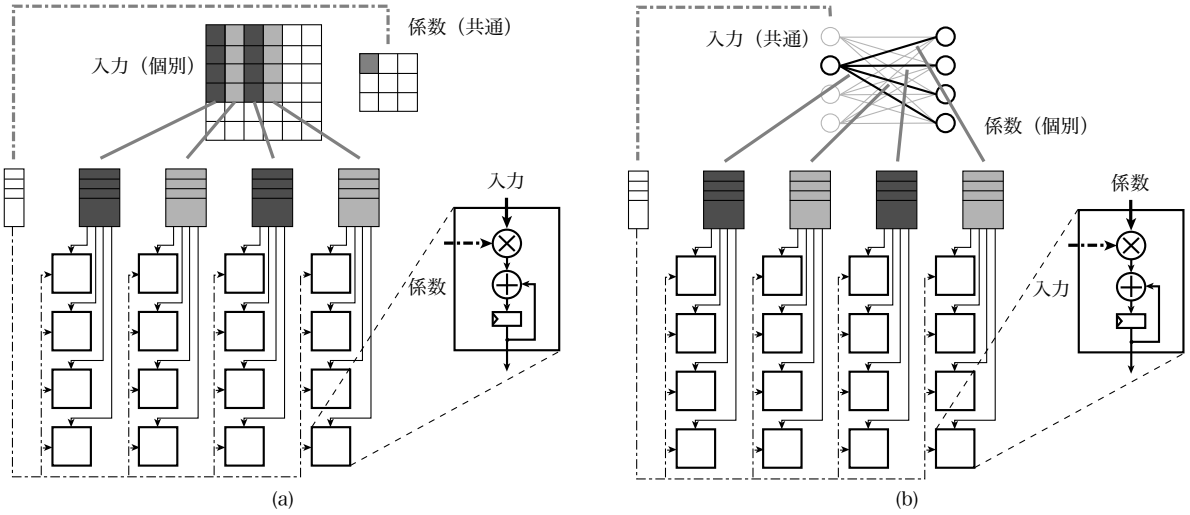

図 3　畳込み層 (a) と全結合層 (b) のデータ配分

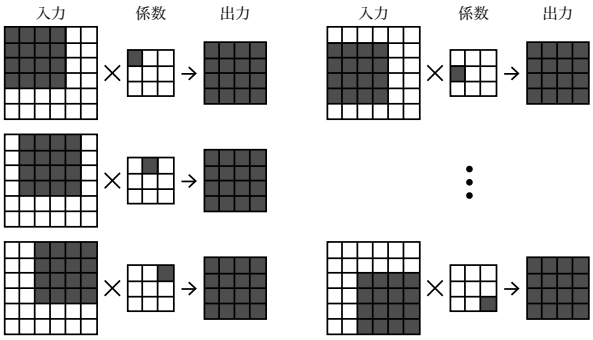

図 4　演算器アレイによる畳込みの基本演算手順

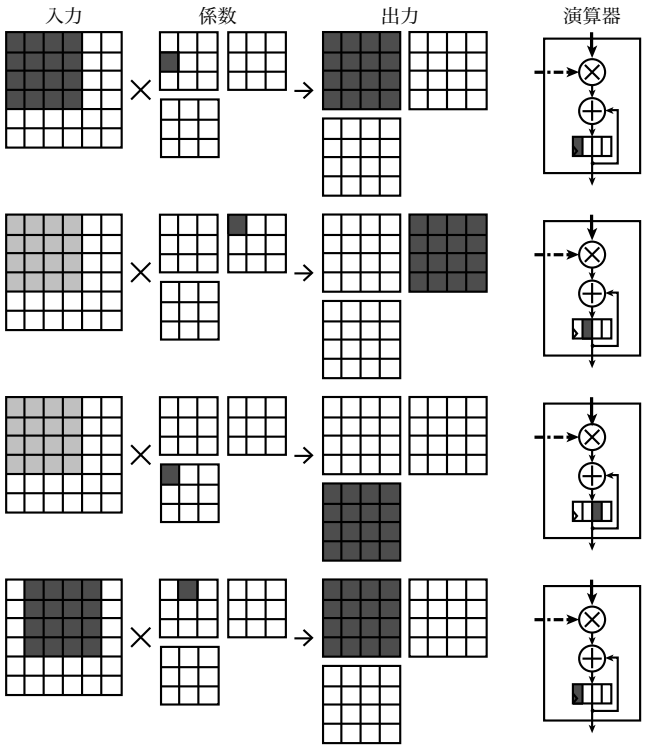

図 5　複数チャネルの入力データ削減

の矩形領域をそのまま演算器アレイにマップすると、係数を一つ選んだ際に各出力要素 (=演算器) に対応する入力平面の領域も矩形となる。その矩形領域を列単位のバッファに置くことで入力要素の個別配布が実現する。

全結合層の処理 (図 3(b)) にあっては全演算器に共通の入力要素を与え、個別にそれぞれの係数を供給する。全結合層では出力は一次元なので機械的に演算器アレイにマップする。入力要素を一つ選んだとき各出力要素 (=演算器) への重み係数は係数行列の１列であるので、機械的に畳込み層と同じ列バッファで個別配布可能であるとわかる。

3.2　畳込み層の処理とデータ流最適化

これまで考えてきた演算器アレイの動作を畳込み層と全結合層それぞれについて検証し、データ流の最適化を試みる。

この演算器アレイによる単チャネルの畳込みの基本演算手順を図 4 に示す。通常、畳込み層の入力と出力は同じ平面サイズの二次元データの複数集合 (チャネル) で成っており (入力画像ならば RGB 等の色空間、中間層では各特徴マップに相当)、複数チャネルの畳込みでは各出力チャネルについて全ての入力チャネルからの影響を考慮するが、ここでは説明のため入出力とも１チャネルで考える。演算器アレイの並列数を $P \times P$、係数のサイズを $H \times H$、ストライドを s とする (図では $P = 4$, $H = 3$, $s = 1$)。１サイクルの演算は、係数を１要素選んで演算器アレイの共通入力へ入力、それに対応する入力平面のサイズ $P \times P$ の部分領域を切り出して各演算器に個別に入力することである。これを毎サイクル順に全 $H \times H$ 個の係数に繰り返すことで畳込みの演算が可能である。この例では出力平面全体を演算器アレイがカバーしているが、出力平面サイズが大きく演算器数が不足する場合には、平面を複数ブロックに分けて時分割処理を行う必要がある。

処理速度と電力効率の観点から、できる限り演算量あたりのデータアクセス量は削減したい。畳込みの演算構造を見ると、データアクセスの規則性から以下の二つのデータ転送量低減の可能性があると気づく。

まずひとつは複数チャネルの演算の入力パターンの時間的局所性である。以下、入力チャネル数を C_i、出力チャネル数を

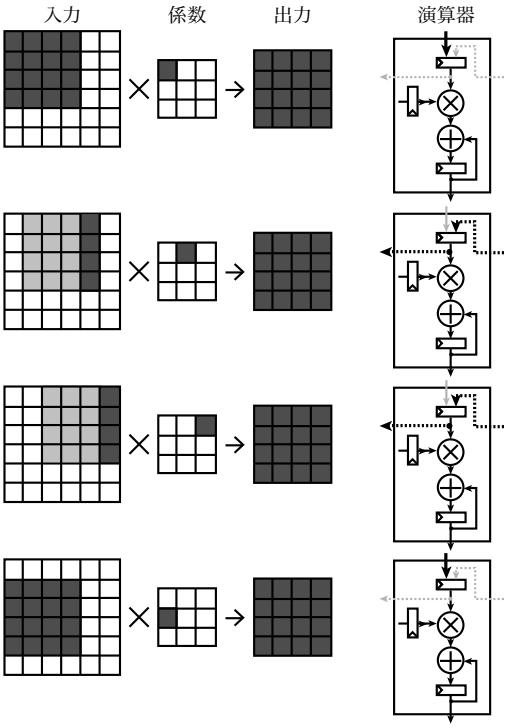

図 6　隣接データ流用による入力データ削減

C_o とする。複数チャネルの畳込みでは各入力チャネルから各出力チャネルへの影響を考えるため、1 入力部分領域は全体で C_o 回参照される。一度ロードされた入力要素を内部で保持したまま複数の出力チャネルに対して積和計算を行うことで、入力要素へのアクセスを緩和することができる。図 5 のように、演算器に入力レジスタ・係数レジスタと T 個のアキュムレータを用意し、一度ロードした入力要素を最大 T 出力チャネルの間で使いまわす (図中では $T = 4$。入力平面に薄灰色で示した領域が新たにロードすることなく流用する領域である)。これによってデータ転送要求量は最良 $1/T$ 倍まで削減できる。

第二に、特に畳込み演算のストライドが $s = 1$ である場合、顕著な空間的局所性も確認できる。同チャネル内で 1 入力要素は $H \times H$ 回、すなわち係数要素の数だけ利用されるのであるが、このうち列方向 (同一係数行内) の H 回は演算が連続している。図 6 に示すように、一つ隣の係数要素が使う入力部分領域は元の領域と 1 列を除いて共通していることがわかる。係数要素行内であれば演算器が次に必要とする入力要素は右隣の演算器が直前サイクルでロードした入力要素と一致する。よって、行内で隣接する演算器の入力レジスタ同士を接続し、可能な場合外部メモリではなく隣接演算器のレジスタからフォワーディングすることで、入力要素の要求量は $1/H$ まで削減することが可能となる (図 6; 入力平面の薄灰色の領域は右隣の演算器からデータを取得し、新規ロードを行わない部分)。

3.3　全結合層の演算

図 7 は全結合層の処理の基本演算手順を示している。入力要素数 N_i、出力要素数 N_o とすると、係数は総計 $N_i \times N_o$ 種類ある。1 演算器はそれぞれ 1 出力要素を担当する。毎サイクル入力要素を順に一つ選んで演算器アレイの共通入力へ入力し、

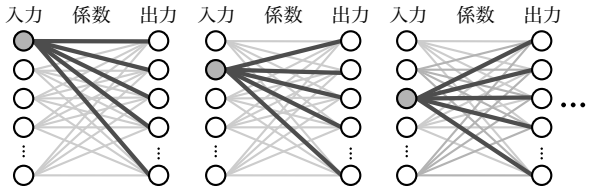

図 7　演算器アレイによる全結合層の基本演算手順

表 1　処理時間 [サイクル] とバンド占有率 [bps]。演算ビット幅 b, クロック周波数 f, 演算器アレイ $P \times P$ 並列, アキュムレータ数 T, フィルタサイズ $H \times H$。

	畳込み層	全結合層
条件	出力 $N_{or}N_{oc}C_o$, 入力 $N_{ir}N_{ic}C_i$	出力 N_o, 入力 N_i
時間	$H^2C_iT\left\lceil\frac{C_o}{T}\right\rceil\left\lceil\frac{N_{or}}{P}\right\rceil\left\lceil\frac{N_{oc}}{P}\right\rceil$	$N_i\left\lceil\frac{N_o}{P^2}\right\rceil$
入力	$bf\frac{(P+2\lfloor\frac{H}{2}\rfloor)N_{ic}}{H^2T\left\lceil\frac{N_{oc}}{P}\right\rceil}$	$bf\left(1+P^2\right)$
出力	$bf\frac{P^2}{H^2C_i}$	$bf\frac{P^2}{N_i}$

各演算器の個別入力には現在の入力要素と対応する係数をそれぞれ入力する。これにより、全入力要素を巡り終わる N_i サイクルで全結合層の処理を行うことができる。

3.4　データ流再構成型並列アーキテクチャ

これまでの議論に即した演算器アレイ型アーキテクチャを図 8 に提案する。乗算アキュムレータを中核としてデータ選択回路を持つのみの単純な演算器を多数並列したホモジニアス並列演算器である。単純な演算器コアは周辺バスの制御による柔軟性に寄与するとともに、並列度を稼ぐ上でも有効である。

この演算器アレイの入力として、各演算器に個別のデータを与える個別入力線と、全演算器に共通のデータを入力する共通入力線の 2 系統を用いる。畳込み層と全結合層とはデータ構造や処理内容が完全に異なるものであるが、出力要素並列の下では必要な入力は共通の 1 データと個別のデータという同一の形が現れる。これを個別入力と共通入力のデータ線にマップすることが可能で、これによって単一のハードウェア構成で畳込みと全結合に両用することが可能となる。個別入力はフォワーディングとの親和性を考慮し、各演算器列の列バッファが列内の演算器にデータ供給する。

多くの CNN で演算時間の大半を占める畳込み層の処理において、外部からのデータ転送を最小に抑え演算効率を向上させることが必要である。前節で検討した二つの局所性に着目し、同一の入力データを複数の出力要素に使う複数アキュムレータと、隣接演算器間の入力データ流用を行うフォワーディング線でデータ転送量を削減する。

個別入力の列バッファと共通入力のブロードキャストバッファの上位に SRAM オンチップバッファが中継するダブルバッファリング構成とした。本システムとホストプロセッサとのデータ中継を担うほか、使用する最大領域をプリフェッチして内部データアクセスの簡略化に貢献する。

4.　提案アーキテクチャの評価

本方式の演算時間、および外部メモリとのデータ転送レート

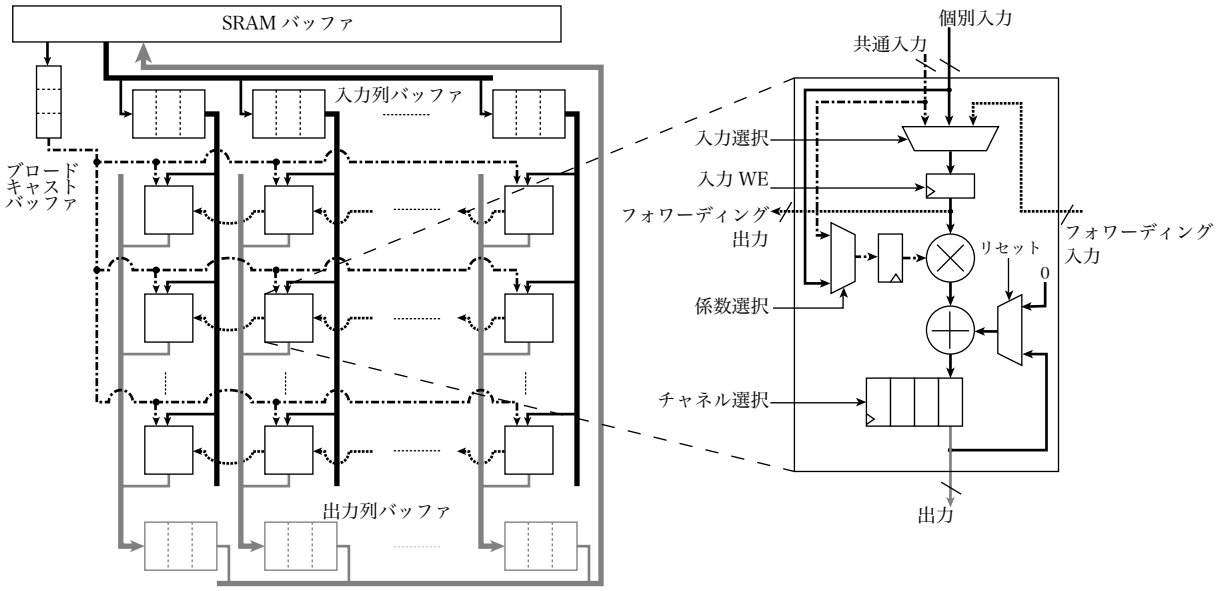

図 8　演算器アレイとバスアーキテクチャ

表 2　AlexNet [5] の処理サイクル数

層	種別	演算サイクル	データレート [Gbps]	並列数 P^2
1	畳込み	313,632	11.4	27^2
2	畳込み	2,764,800	11.4	27^2
3	畳込み	884,736	43.5	27^2
4	畳込み	663,552	43.5	27^2
5	畳込み	442,368	20.9	27^2
6	全結合	735,488	1004.0	250
7	全結合	69,632	1004.0	250
8	全結合	16,384	1004.0	250
合計		5,890,592		

の評価を行った (表 1)。

畳込み層の処理では出力と入力の全チャネルの組合わせについて全ての係数を巡る。出力平面サイズが演算器数より大きいときはブロックに分割して処理を行うので、処理時間は

$$H^2T\left\lceil\frac{C_o}{T}\right\rceil C_i\left\lceil\frac{N_{or}}{P}\right\rceil\left\lceil\frac{N_{oc}}{P}\right\rceil \quad [\text{サイクル}]$$

である。データ転送量は 1 行ブロックの処理で必要となる全領域を SRAM に転送するため、

$$\left(P+2\left\lfloor\frac{H}{2}\right\rfloor\right)N_{ic}\left\lceil\frac{N_{or}}{P}\right\rceil\left\lceil\frac{C_o}{T}\right\rceil \quad [\text{データ}]$$

である。よって畳込みの入力平均バンド占有率は、演算ビット幅を b、動作周波数を f として

$$bf\frac{\left(P+2\left\lfloor\frac{H}{2}\right\rfloor\right)N_{ic}}{H^2T\left\lceil\frac{N_{oc}}{P}\right\rceil} \quad [\text{bps}]$$

である。SRAM バッファの容量が許せば演算器単体のデータ転送削減より上位でデータキャッシュとして働く。

表 2 に、CNN の標準的なベンチマークである AlexNet [5] の本方式による処理サイクル数とデータレート (バンド占有率) の試算結果を示す。なお想定システムの最大バンド幅は 2 Tbps である。並列数は畳込み層では $P = 27$ ($P^2 = 729$ コア並列)、全結合層では $P^2 = 250$ とした。表 2 の合計 5,890,592 サイクルは 200 MHz では 29.5 ms であるので、本システムは AlexNet を毎秒 33 フレーム処理できることがわかる。また表 3 に文献 [4] との畳込み層の処理時間の比較を示す。

表 3　既存研究との比較。バッチサイズ 4 の AlexNet [5] の処理時間。

層	提案手法	既存手法 [4]
1	6.3 ms	20.9 ms
2	55.3 ms	41.9 ms
3	17.7 ms	23.6 ms
4	13.3 ms	18.4 ms
5	8.8 ms	10.5 ms
合計	101.4 ms	115.3 ms

並列度 P はダイサイズのほかバンド幅に律される。表 2 から畳込み層においては想定システムの 2 Tbps に十分収まっているが、畳込み層で顕著なデータの相関・局所性が全結合層には全くないためバンド幅律速を強く受け、畳込み層処理時に比して並列度を限定せざるを得ないことがわかる (表 2 で 1 Tbps 程度におさめるため $P^2 = 250$ とした)。しかしその場合でも残る並列度と単純なアレイ構造によって何ら演算オーバヘッドを生じることなく処理を行うことが可能であると思われる (表 2 でも全結合層の処理時間の影響は支配的でない)。

以上から、共通入力に逐次的に係数 (または入力) を流しながら個別入力で並列的に入力 (係数) を与えて並列演算を行う本提案が、標準的な規模の CNN のアクセラレーションに十分実用可能であると考える。

問題規模と時間特性・リソース特性との関係を図 9 および図 10 に掲げる。ともに並列度は $P = 27$ (27^2 並列) である。表 1 での検証の通り、処理時間は出力平面面積 ($N_{or}N_{oc}$) にほぼ比例し (N_{ic} は N_{oc} に正比例)、出力チャネルに比例する。一方でバンド占有率は出力サイズによらずほぼ一定となっている。問題規模が大きくなると処理時間は増大するが、バンド幅等の必

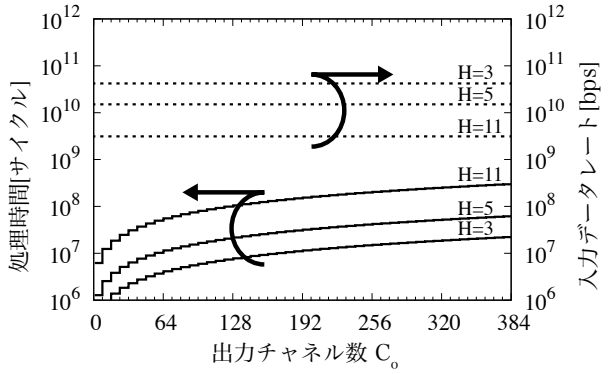

図 9　出力チャネル数と処理時間・バンド占有率 ($C_i = 256, N_{or} = N_{oc} = 128$)

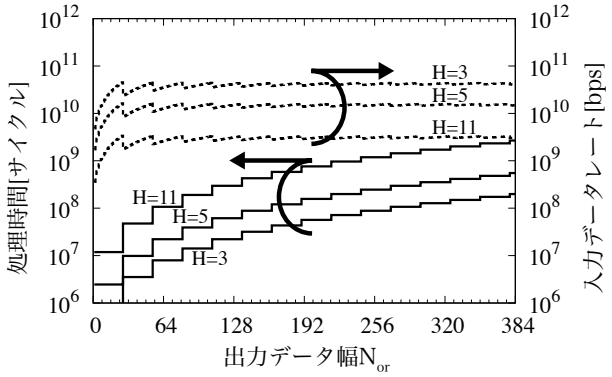

図 10　出力平面サイズと処理時間・バンド占有率 ($C_i = 256, C_o = 384$)

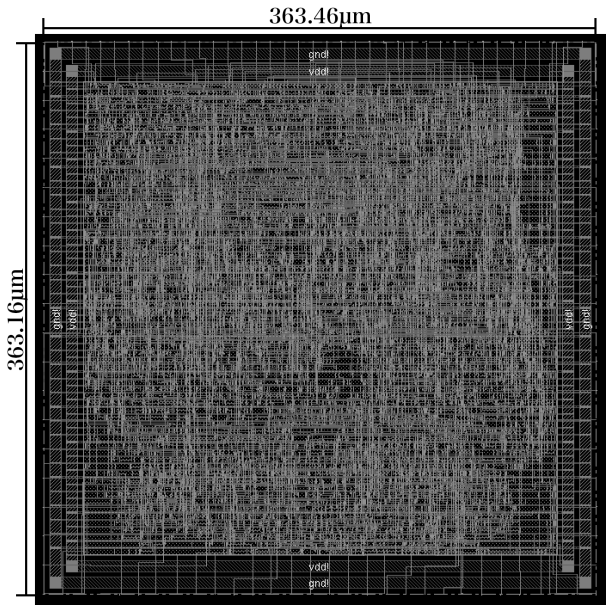

図 11　1 演算器のレイアウト結果

要リソースには大きく影響しないので、問題規模に対する可用性は保たれている。

図 11 と表 4 に本提案の演算器 1 個を配置配線した結果を示す。これより 180 nm プロセスで 1 cm 角チップに実装したとして 27×27 並列の演算器アレイの構成が実現可能であると考える。

表 4　性 能 諸 元

プロセス	180 nm
面積	363×363 μm^2
最高動作周波数	200 MHz
演算精度	20 ビット固定小数点
ゲートカウント	7,738

5. ま と め

CNN の二大要素たる畳込み層と全結合層の両方を単一ハードウェアで効率的に処理できるアーキテクチャを検討した。

畳込み層と全結合層の演算の特質について述べ、それらの類似点と相違点を説明した。二つの演算はデータのトポロジーが全く異なっているが、根幹の演算は双方とも積和演算であり、乗算アキュムレータを原型とする単純な積和演算器のアレイ化で両方の処理に対応可能であることを示した。両処理の演算手順の類似点から、演算器アレイの全演算器に単一の入力を与える共通入力と、各演算器に固有の入力を供給する個別入力を用意し、それらを適時制御することで処理能力と効率を落とさずに同じ演算器アレイの構造が利用可能である。

畳込み層の演算に際し、演算のデータ構造の特徴を活かし一度読み込んだデータを次の演算に再利用することで転送データ量を低減しながら効率的に処理する方式を提案した。全結合層の処理についても個別入力と共通入力の 2 系統の入力の制御により畳込み層と同じ演算器アレイを用いて十分に実効性能を引き出せることは確認できたが、畳込み層と異なりデータの流用が不可能なのでデータ転送のバンド幅ボトルネックを脱することができていない。今後はデータ供給の方式を見直し、全結合層の処理においても並列度を下げることなく対応できる方式を模索すべきと考える。

また演算器単体での論理合成・配置配線の結果、面積も想定通りであり 200 MHz 駆動が可能であるとの見積もりを得た。今後の展望として、これを実際に並列化して実動するシステムを構築し、その動作の評価を行いたい。

文　献

[1] “NVIDIA CUDNN - GPU Accelerated Deep Learning,” https://developer.nvidia.com/cudnn

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

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

[4] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, “Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks,” *2016 IEEE International Solid-State Circuits Conference*, pp. 262-263, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Advances in Neural Information Processing Systems 25*, pp. 1097-1105, 2012.