2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture

Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks

Yu-Hsin Chen*, Joel Emer*† and Vivienne Sze*

*EECS, MIT
Cambridge, MA 02139

†NVIDIA Research, NVIDIA
Westford, MA 01886

*{yhchen, jsemer, sze}@mit.edu

2020年06月10日

発表者：王志辰

一、序論

- 深層畳み込みニューラルネットワーク(CNN)では、フィルタウェイトの保存に最大数百メガバイトの空間、入力ピクセルあたり30k～600kの演算が必要である。

- 大規模化のネットワークは、基礎となる処理ハードウェアにスループットとエネルギー効率の両方の課題がある。

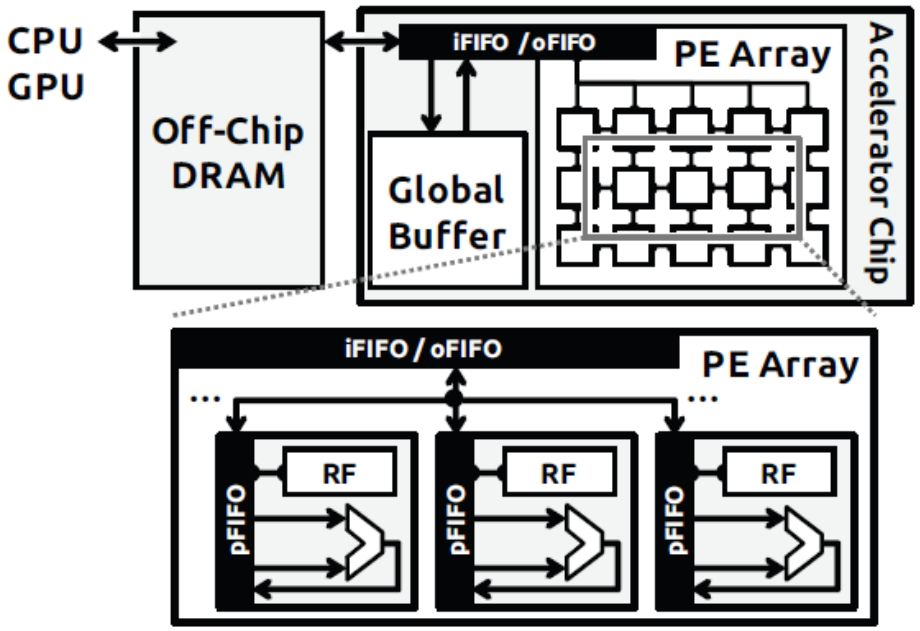
一、序論

- 要約すると、本研究の主な貢献は:
  - 先行研究の既存のCNNデータフローを分類する分類法。
  - スループットとエネルギー効率のために最適化された「行定常」と呼ばれる新しいCNNデータフローに基づく空間アーキテクチャ。「行定常」は畳み込み層と全結合層の両方で動作し、ストレージ階層内のあらゆるタイプのデータ移動を最適化する。
  - 同じハードウェア制約の下で、異なるCNNデータフローのエネルギー効率を定量化できる解析フレームワーク。また、各データフローに対して最もエネルギー効率の良いマッピングを探索することができる。
  - 様々なCNNデータフローについて、データ移動に関連するエネルギーコストと、異なるタイプのデータ再利用の影響を比較分析した結果を提示する。

二、空間アーキテクチャ

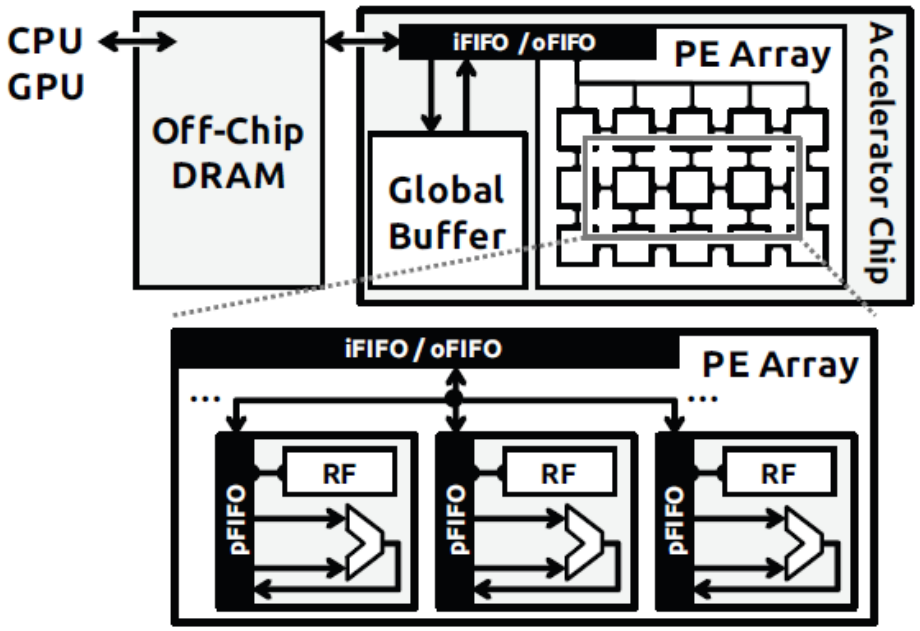
- 空間アーキテクチャ(Spatial Architecture)は、比較的単純な処理エンジン(Processing Engine)の配列間の直接通信を使用して、高い計算並列性を利用できるアクセラレータである。
- 空間アーキテクチャ(SA)には2つの種類がある：
  - ALUスタイルの処理エンジンをオンチップネットワークで接続したタイル状のアレイで構成される粗視化SA。
  - 通常はFPGAの形をしているファイングレインSA。
- 粗視化されたSAは現在CNNアクセラレータの実装として非常に人気のある選択肢である。
- しかし、どちらのタイプのSAでも、CNNデータフローをSAに正確にマッピングすることが課題となる。

二、空間アーキテクチャ



- 本論文で使用するCNN処理用アクセラレータシステムはSAアクセラレータとオフチップDRAMから構成されている。
  - 入力はCPUやGPUからDRAMにオフロードされ、 SAで処理されます。
  - そして、出力はDRAMに書き戻され、メインプロセッサによってさらに解釈される。

二、空間アーキテクチャ



- SAはグローバルバッファとPEのアレイで構成されている：
  - グローバルバッファは、入力データの再利用やDRAMアクセスのレイテンシを隠したり、中間データの保存に使用することができる。
  - PE には、ALU データパス、レジスタファイル (RF)、ALUのトラフィックを制御するための PE FIFOが含まれている。

- 全体的には、 4つのレベルのストレージ階層でデータアクセスが可能である：DRAM、グローバルバッファ、アレイ（PE間通信）、RF。
- 異なるレベルからデータにアクセスすると、エネルギーコストも異なり、DRAMで最もコストが高く、RFで最もコストが低くなる。

三、 CNN処理（CONVとFC層）の課題

- データ処理：
  - すべての入力をDRAMから直接読み出すためには、高い帯域幅が必要で、高いエネルギー消費が発生する（入力データ再利用で解決）：
    - 畳み込み再利用（少量のユニークな入力データを多くの演算で共有する）。
    - フィルタの重みの再利用。
    - Ifmap（入力特徴図）の再利用。
  - かなりの量の中間データ、すなわち部分和（psum）が同時に生成されるため、ストレージの圧力がかかり、処理されない場合、追加のメモリR/Wエネルギーを消費する：
    - 適切な操作スケジューリングによって処理され、生成された psum を可能な限り早く削減する。
  - 高いスループットとエネルギー効率を達成するために、基礎となるCNNデータフローは、入力データの再利用とpsum蓄積スケジューリングの両方を同時に考慮する必要がある。

三、CNN処理（CONVとFC層）の課題

- 適応処理：
  - 同じCNNモデル内であっても、各層は異なる形状の構成を持っている。したがって、ハードウェア・アーキテクチャは、特定の形状だけを処理することはできない。その代わりに、データフローは異なる形状に対して効率的でなければならない。ハードウェアアーキテクチャも、効率的なデータフローに動的にマッピングできるようにプログラム可能でなければならない。
- CNNが主流になる前から、画像信号処理（ISP）での高効率畳み込みの研究が行われていた。しかし、それらは2つの理由からCNN処理には直接適用できない：
  - CNNのフィルタ重みは，処理システムで固定されているのではなく，訓練によって得られるものである。
  - ISP技術は主に2次元畳み込みのために開発されている。

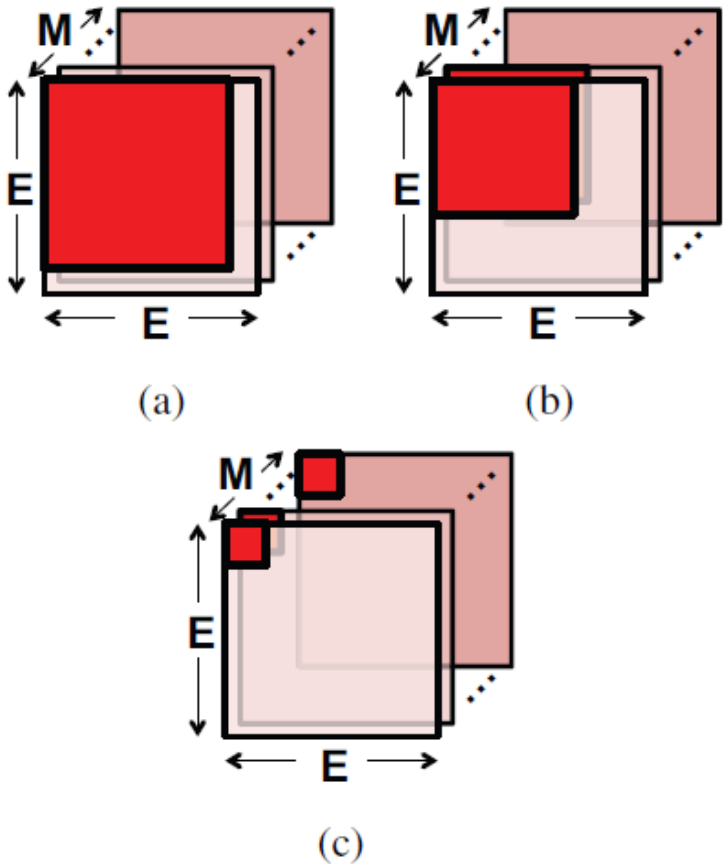
四、既存のCNNデータフロー

- ウェイト定常（WS）データフロー
  - 定義：各フィルタウェイトは、畳み込み再利用とフィルタ再利用を最大化するために、RF内で静止したままである。
  - 処理方式：同じフィルタとチャネルからのR×R重みは、R×R PEの領域にレイアウトされ、静止したままである。
  - ハードウェアの使用法：RFは、固定されたフィルタ重みを格納するために使用される。静止した重みを最大に再利用する操作スケジューリングのため、psum は常にすぐに還元可能とは限らず、一時的にグローバルバッファに格納されます。

四、既存のCNNデータフロー

- 出力定常(OS)データフロー
  - 定義：各ofmap （出力特徴図）ピクセルの蓄積はPE内で静止したままである。 psum蓄積コストを最小化するために、同じRFに蓄積される。
  - 処理方式：このタイプのデータフローでは、PEアレイの空間を利用して4次元ofmapの領域を一度に処理する。高次元空間から領域を出力選択するには、2つの選択肢がある：
    - 複数のofmapチャネル(MOC)と単一のofmapチャネル(SOC)
    - 複数のofmapプレーンピクセル(MOP)と単一のofmapプレーンピクセル(SOP)

四、既存のCNNデータフロー



- 出力定常(OS)データフロー
  - 処理方式：これにより、実用的な3つのOSデータフローのサブカテゴリが作成される：
    - SOC-MOP：主にCONV層に使用され、一度に1つの平面のmapを処理する。
    - MOC-MOP：同一平面内に複数の画素を持つ複数のofmapプレーンを一度に処理する。
    - MOC-SOP：主にFCレイヤで使用される。これは、複数のofmapチャネルを処理するが、チャネル内のピクセルは一度に1つだけである。
  - すべての追加入力データの再利用は、アレイレベル、すなわちPE間通信で利用されている。RF レベルでは psum の蓄積のみを処理する。
- ハードウェアの使用法：すべての OS データフローは、 psum ストレージに RF を使用して定常蓄積を実現する。

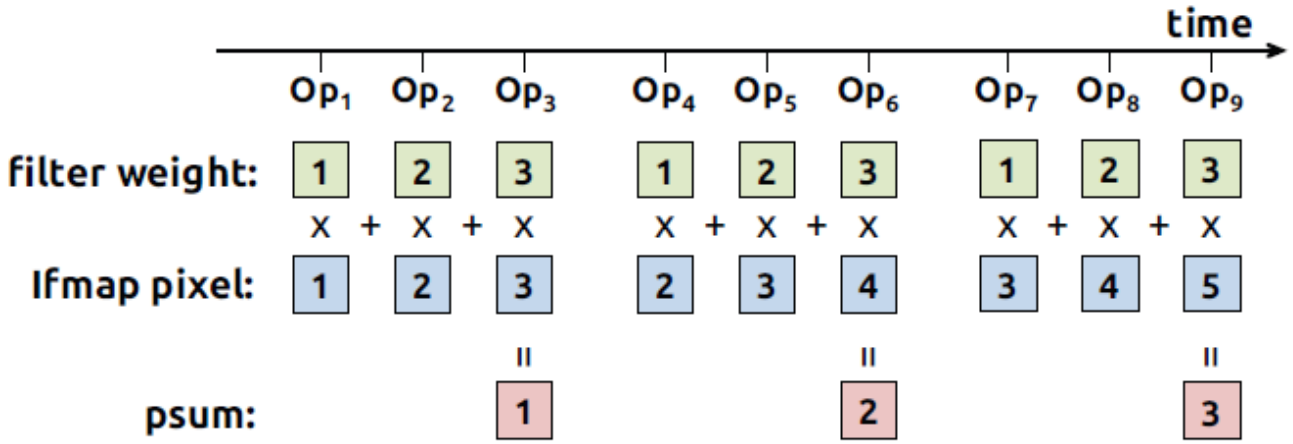
四、既存のCNNデータフロー

- ローカル再利用なし（NLR）データフロー
  - 定義：NLR データフローには大きく分けて 2 つの特徴がある：
    - RFレベルでのデータ再利用を利用しないこと。
    - ifmap再利用とpsum蓄積にはPE間通信を利用していること。
  - 処理方式：NLR は PE 配列を PE のグループに分割します。同じグループ内のPEは、同じ入力チャンネルから同じイフマップピクセルを異なるフィルタウェイトで読み取る。異なる PE グループは、異なる入力チャンネルの ifmap ピクセルとフィルタの重みを読み取る。生成されたpsumは、アレイ全体のPEグループに渡って蓄積される。
  - ハードウェアの使用法：必要とされるRFストレージはない。PE 配列は単に ALU データパスで構成されているため、グローバルバッファのための大きな領域を残しています。これは、再利用のための入力データだけでなく、psumを保存するためにも使用される。

五、行定常(RS)データフロー

- 1次元畳み込みプリミティブ
  - 高次元の畳み込みを1次元の畳み込みプリミティブに分解し，並列に実行できるようにする。各プリミティブは1行のフィルタ重みと1行のifmapピクセルを操作し、1行のpsumを生成する。異なるプリミティブからの psum をさらに蓄積して ofmap ピクセルを生成する。 1次元畳み込みへの入力は、ストレージ階層、例えば、グローバルバッファまたはDRAMから来る。
  - 各プリミティブは、処理のために１つのPEにマッピングされているため、各行ペアの計算はPE内で静止したままであり、これにより、RFレベルでフィルタ重みとifmapピクセルの畳み込み再利用が行われる。
  - しかし、畳み込み全体は通常数十万のプリミティブを含むため、すべてのプリミティブをPE配列に正確にマッピングすることは、エネルギー効率に大きな影響を与えることになる。

五、行定常(RS)データフロー

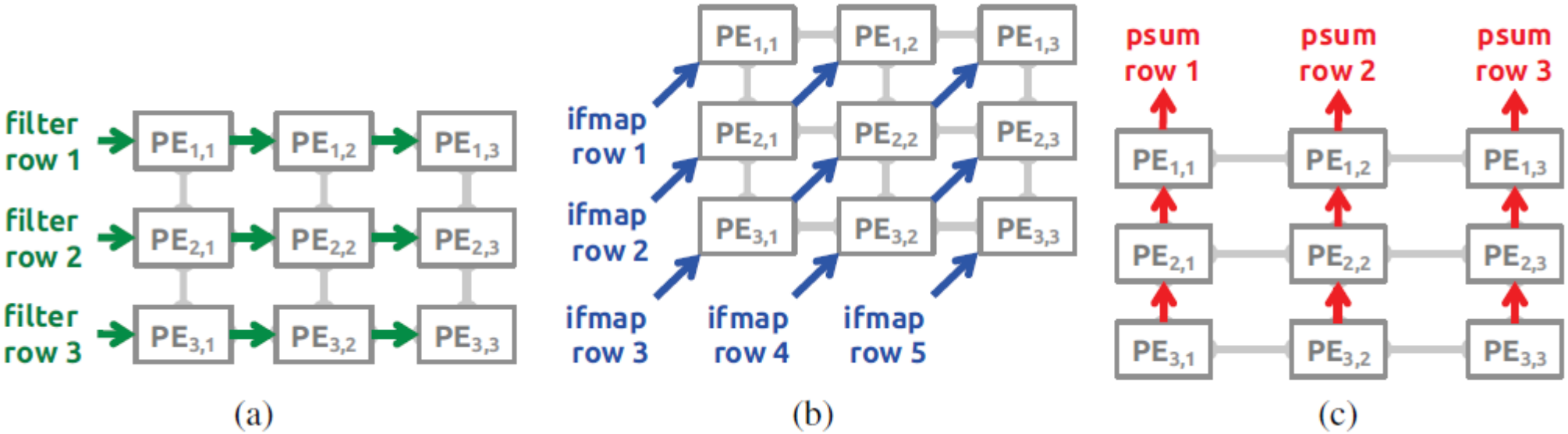


PEにおける1次元畳み込みプリミティブの処理

五、行定常(RS)データフロー

- 二段階のプリミティブマッピング
  - 論理マッピング：1 次元畳み込みプリミティブの数と同じサイズで、通常はハードウェアの物理 PE 配列よりもはるかに大きい論理 PE 配列にプリミティブを配置する。
    - 各1 次元プリミティブは、最初に論理PE 配列の1 つの論理PE にマッピングされる。論理 PEたちを論理 PE 集合としてグループ化する。
    - 各フィルタ行と ifmap 行がそれぞれ水平方向と斜め方向に再利用され、各行の psum が垂直方向に蓄積された論理 PE セットである。

五、行定常(RS)データフロー



2次元畳み込みを処理するための論理PEセット内のデータフロー：

(a) フィルタ重みの行は、水平方向に PE 間で再利用される。

(b) ifmap pixelの行は、PEの対角線上で再利用される。

(c) psum の行は、垂直方向に PE 間で蓄積される。

五、行定常(RS)データフロー

- 二段階のプリミティブマッピング
  - 物理マッピング：論理 PE 配列を折り畳み、物理 PE 配列に収まるようにする。折り畳みは、異なる論理 PE からの複数の 1 次元畳み込みプリミティブを同じ物理 PE にマッピングして実行することである。
    - 異なるセットの同じ位置にある複数の論理PEを単一の物理PE上に折り畳むことで、RFレベルでの入力データの再利用とpsumの蓄積を可能にする。
    - 対応する 1 次元畳み込みプリミティブは、同じ物理 PE 上でインターリーブされた形で実行される。
  - 物理PEアレイは、多数の論理PEセットを処理することができて、処理パスと呼ばれる。しかし、一つの処理パスは足りないかもしれなくて、2段階の折り畳みが必要とされる。そして、異なる処理パスは、物理的なPEアレイ全体で順次実行される。グローバルバッファーは、入力データの再利用をさらに活用し、パス間のpsumを保存するために使用される。

五、行定常(RS)データフロー

- エネルギー効率の高いデータ処理
  - エネルギー効率を最大化するために、RSデータフローはストレージ階層の使用量を最大化することで、あらゆるタイプのデータ移動を最適化するように構築されている：
    - RF：第1段階の折り畳み後、PE内で複数の1次元畳み込みプリミティブを実行することで、RFはあらゆるタイプのデータの動きを最大限に利用する。具体的には、各プリミティブの計算内での畳み込み再利用、折り畳まれたプリミティブ間での入力データの共有によるフィルタ再利用やifmap再利用、各プリミティブ内やプリミティブ間でのpsum蓄積などがある。
    - 配列（PE間通信）：畳み込み再利用は各集合内に存在し、このレベルまでは完全に枯渇している。フィルタ再利用やifmap再利用は実現できる。 psum蓄積は、各セット内だけでなく、空間的にマップされたセット間でも行われる。
    - グローバルバッファ：サイズにもよりますが、グローバルバッファは、 2段階の折り畳後にRFレベルとアレイレベルに残ったフィルタ再利用、ifmap再利用、psum蓄積の残りの部分を利用するために使用される。

五、行定常(RS)データフロー

- 異なるレイヤータイプのサポート
  - RSデータフローは、CONV層の高次元の畳み込みを処理するために設計されていますが、他の2つの層も自然にサポートしている：
    - FC層：全結合層の計算は畳み込み層と同じであるが、畳み込みデータの再利用はない。RS データフローはすべてのタイプのデータ移動を利用するので、ストレージ階層の各レベルでフィルタ再利用、ifmap 再利用、および psum 蓄積使用することができる。SOC-MOPとMOC-SOP OSのデータフローのように、異なるデータフローを切り替える必要はない。
    - POOL層：RSデータフローでは、N=M=C=1と仮定して各fmapプレーンを個別に実行することで、POOL層の処理も可能である。

五、行定常(RS)データフロー

- 当アーキテクチャの他の特徴
  - 当アーキテクチャでは、以下の方式でスパース性を利用することもできる：
    - ゼロ以外の値のみに対してデータの読み取りと計算を実行すること。
    - データの移動を減らすためにデータを圧縮すること。
  - これにより、本論文で紹介する効率的なデータフローに加えて、さらなるエネルギー節約がもたらされる。

六、実験方法