

基于线性脉动阵列的卷积神经网络计算优化与性能分析

刘勤让, 刘崇阳, 周俊, 王孝龙

(国家数字交换系统工程技术研究中心, 河南 郑州 450002)

摘要: 针对大部分 FPGA 端上的卷积神经网络 (CNN, convolutional neural network) 加速器设计未能有效利用稀疏性的问题, 从带宽和能量消耗方面考虑, 提出了基于线性脉动阵列的 2 种改进的 CNN 计算优化方案。首先, 卷积转化为矩阵相乘形式以利用稀疏性; 其次, 为解决传统的并行矩阵乘法器存在较大 I/O 需求的问题, 采用线性脉动阵列改进设计; 最后, 对比分析了传统的并行矩阵乘法器和 2 种改进的线性脉动阵列用于 CNN 加速的利弊。理论证明及分析表明, 与并行矩阵乘法器相比, 2 种改进的线性脉动阵列都充分利用了稀疏性, 具有能量消耗少、I/O 带宽占用少的优势。

关键词: 线性脉动阵列; 卷积神经网络; 稀疏性; I/O 带宽; 性能分析

中图分类号: TP183

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018100

Based on linear systolic array for convolutional neural network's calculation optimization and performance analysis

LIU Qinrang, LIU Chongyang, ZHOU Jun, WANG Xiaolong

(National Digital Switching System Engineering and Technological R&D Center, Zhengzhou 450002, China)

Abstract: Concerning the issue that the convolutional neural network (CNN) accelerator design on most FPGA ends fails to effectively use the sparsity and considering both bandwidth and energy consumption, two improved CNN calculation optimization strategies based on linear systolic array architecture are proposed. Firstly, convolution is transformed into matrix multiplication to take advantage of sparsity. Secondly, in order to solve the problem of large I/O demand in traditional parallel matrix multiplier, linear systolic array is used to improve the design. Finally, a CNN acceleration comparative analysis of the advantages and disadvantages between parallel matrix multiplier and two improved linear systolic arrays is presented. Theoretical proof and analysis show that compared with the parallel matrix multiplier, the two improved linear systolic arrays make full use of sparsity, and have the advantages of less energy consumption and less I/O bandwidth occupation.

Key words: linear systolic array, convolutional neural network, sparsity, I/O bandwidth, performance analysis

收稿日期: 2018-08-10; 修回日期: 2018-10-29

通信作者: 刘崇阳, zmylmh1@163.com

基金项目: 国家科技重大专项基金资助项目 (No.2016ZX01012101); 国家自然科学基金资助项目 (No.61572520); 国家自然科学基金创新研究群体资助项目 (No.61521003)

Foundation Items: The National Science Technology Major Project of China (No.2016ZX01012101), The National Natural Science Foundation of China (No.61572520), The National Natural Science Foundation Innovation Group Project of China (No.61521003)

1 引言

CNN 是一种前馈神经网络,随着人工智能的发展,深度学习中的 CNN 已成为物体识别的主要方法,广泛用于计算机视觉领域。但 CNN 是一个计算密集型和存储密集型的网络^[1-2],为了面向更复杂的任务,CNN 的规模正向更深、更复杂方向发展。例如,文献[3]指出:“标准”的 CNN 结构层次太少,只有神经元、神经网络层,要把每一层的神经元组合起来形成组并装到“舱”(capsule)中完成大量的内部计算。

CNN 规模大和计算量大的特点导致移动端上部署变得很困难,规模太大可以压缩^[4-5],计算量太大就须进行加速。如今,CNN 嵌入式端加速主要集中于 FPGA^[6]。例如,文献[7]结合卷积计算特点用多个二维卷积器实现三维卷积操作,充分利用 FPGA 上资源实现高速计算。文献[8]首先考虑 FPGA 资源限制(如片上存储、寄存器、计算资源和外部存储带宽),之后为给定的 CNN 模型探索出基于 OpenCL 的最大化吞吐量的 FPGA 加速器。文献[9]在 FPGA 上部署 Winograd 算法进行卷积加速。但以上利用 FPGA 的加速方法都未能利用卷积计算中的稀疏性。利用卷积计算中稀疏性的加速方法有文献[10],采用行固定的处理数据方式来利用稀疏性,当数据通道的输入为 0 值时直接不做处理。文献[11]利用并行矩阵乘法器完成卷积计算并充分利用了参数稀疏性。文献[12]把 Winograd 算法和稀疏性进行结合减少卷积操作,改变 Winograd 作用域来利用稀疏性,Winograd 算法计算快但对数据输入输出要求高,运用到 FPGA 上时存取速度成了限制因素。

本文以文献[13-14]中的线性脉动阵列为基础,对处理单元(PE)做细微的“不传不运算操作”控制,以实现卷积计算稀疏性的利用。本文采用的线性脉动阵列主要有 3 个改进:①针对稀疏性的不传不运算操作;②针对卷积核矩阵的部分传输操作;③针对输出结果时的缓存减少策略(针对单输出线性脉动阵列)。本文主要从计算循环周期、存取操作类别和存取操作次数以及资源耗费情况来对比分析和证明,分析表明:改进

后的 2 种线性脉动阵列不仅充分利用了稀疏性,而且不同程度地减少了 I/O 能耗需求。

2 问题描述

2.1 CNN 稀疏性特点

CNN 卷积层的卷积计算有一个激活函数,常用的激活函数 Relu 示意如图 1 所示。

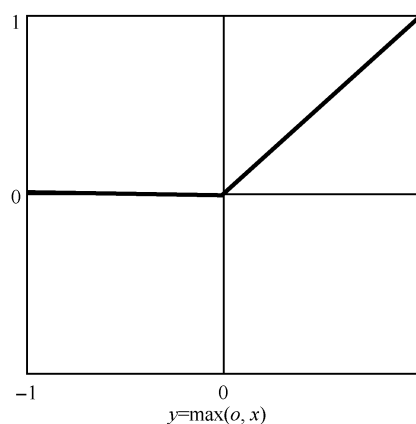


图 1 Relu 示意

经过 Relu 激活函数作用后,小于等于 0 的都变为 0 值,所以输出的特征图有很多 0 值存在。本文基于 Caffe 深度学习框架^[15]进行实验,发现 Alexnet、Googlenet、Caffenet 和 Vgg 等卷积神经网络中都存在较大程度的稀疏性,如 Alexnet 各个卷积层的 0 值比例如图 2 所示。

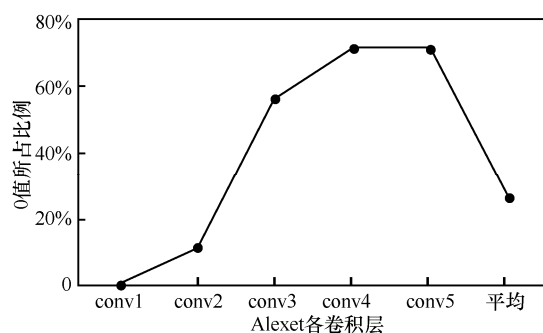


图 2 Alexnet 卷积层 0 值比例

为了利用稀疏性进行加速,本文把卷积过程转化为矩阵相乘,转化的方法是文献[16]中所用的 Im2col,如图 3 所示。该方法存储空间要求增大但计算量(乘法和加法操作)没有变化,若能有效利用稀疏性使 0 值不参与运算,可以在速度上明显改进^[17]。

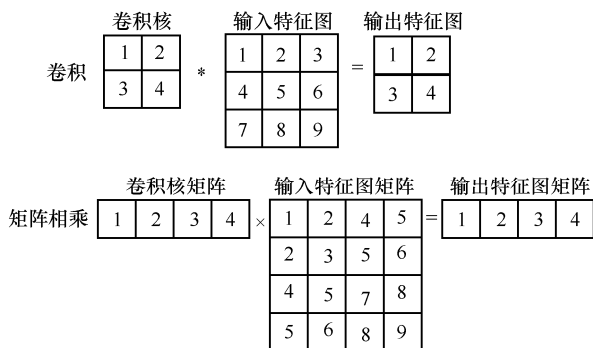


图3 卷积转换为矩阵相乘

2.2 并行矩阵乘法器用于 CNN 加速

CNN 卷积转换为矩阵相乘的方式后,可以采用 FPGA 上并行矩阵乘法器进行加速^[11-18],如图 4 所示,改进为一个单列的 P 个 PE 的结构,在进行 $C = A \times B$ (维度均为 $n \times n$) 的矩阵乘法时, A 每次送入一个元素进行乘加运算,并且被 P 个 PE 共享, A 按照先行后列 (如第一次送第 1 行第 1 列的元素,第二次送第 1 行第 2 列的元素) 的顺序依次送入矩阵乘法器中; B 按行的形式每次输入 P 个数据 (如第一次送第 1 行的前 P 个元素,第二次送第 1 行接下来的 P 个元素) 到矩阵乘法器,并与 A 中同一个数据进行乘加操作。

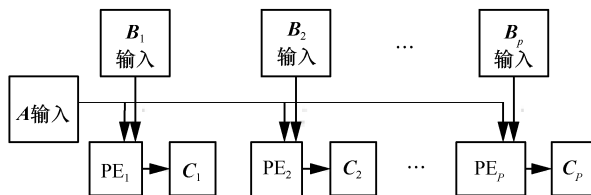


图4 文献[11]的并行矩阵乘法器单列结构

上述方式在 A 每次输入时进行判断,为 0 时不传进去进行乘加操作,因此该方式能很好地利用 CNN 的稀疏性。但存在两点不足:① 上述并行矩阵乘法器每个时刻 (对于 A 输入为非 0 值时) 都需要往每个 PE 中传入具体的值,一个并行矩阵乘法器就要使用 $2P$ 个输入端口,带宽需求非常高;② 该并行矩阵乘法器计算时 A 循环完一行就需要把 B 所有数据输入一遍,由于不存在复用关系, B 矩阵每个数据都要从片上 Cache 读取。从表 1 可以看出^[19],片内 SRAM Cache 读取耗能是寄存器耗能的 10 倍,能量消耗较多,若能让输入数据在多个 PE 之间复用并

采用能量消耗较低的寄存器间的传输 (本文采用线性脉动阵列),则可降低耗能。

表 1 45 nm CMOS 相关操作能量消耗

操作类型	能量消耗/pJ	能量消耗比例
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	640	6 400

3 改进的线性脉动阵列

为了弥补传统并行矩阵乘法器的缺点,并充分利用 CNN 计算中的稀疏性特点进行加速,本文基于文献[13-14]提出的线性脉动阵列进行了两方面改进,分别是单输出线性脉动阵列和多输出线性脉动阵列,具体如图 5~图 8 所示。

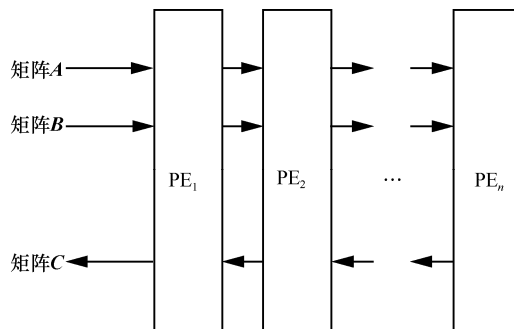


图5 单输出线性脉动阵列

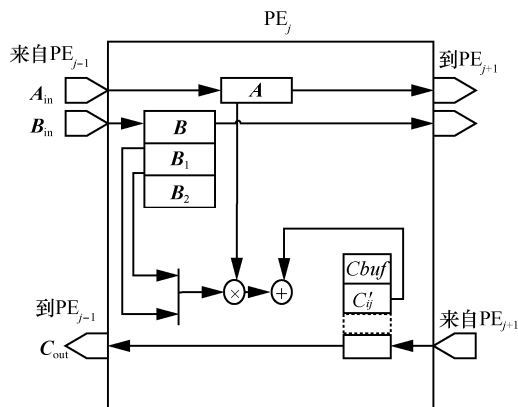


图6 PE 内部结构示意图

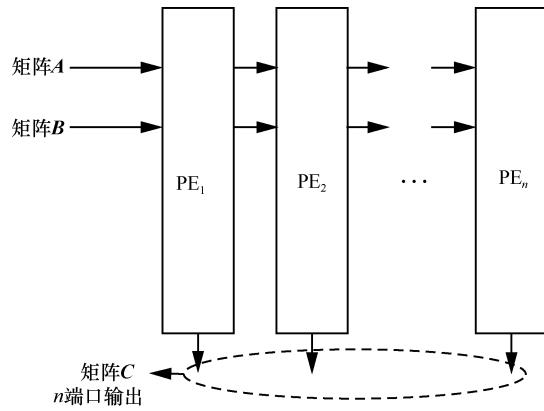


图 7 多输出线性脉动阵列

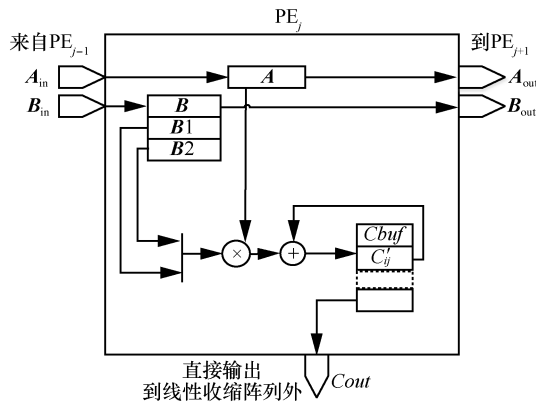


图 8 PE 内部结构示意图

单输出线性脉动阵列和多输出线性脉动阵列仅在输出时有差别，本文以单输出线性脉动阵列为例说明具体计算方式。对于 $n \times n$ 的矩阵，改进线性脉动阵列需要 n 个处理单元 (PE)。矩阵乘法时，矩阵 A 以列的形式输入， B 以行的形式输入， PE_j 完成 C 矩阵第 j 列的计算以及存储。 B 矩阵中的值 B_{xj} 到达 PE_j 后会被截取到其内部寄存器（即 B_{xj} 依次经过 $PE_1, PE_2, \dots, PE_{j-1}, PE_j$ ），并保留一段时间。下面说明需要另外 2 个寄存器存储输入矩阵 B 的元素值，并且 B 矩阵要领先于 A 矩阵 n 个周期传输。

PE_j 在计算 $C'_{ij} = C'_{ij} + A_{ix} \times B_{xj}$ 时，有 2 个必要要求：首先，对于任意的 $1 \leq i \leq n$ ， B_{xj} 应早于 A_{ix} 到达 PE_j ；其次，等到 $A_{1x}, A_{2x} \dots A_{nx}$ 都通过 PE_j 后才不用继续保留 B_{xj} 。对于第一个要求，一个极限情况，如 B_{1n} 早于 A_{11} 到达 PE_n ，对于 B_{1n} 从 PE_1 传输到 PE_n 时需要 $2n$ 个周期， A_{11} 传输到 PE_n 需要 n 个周期，为保证满足要求， A 开始传的时刻要落后 B 矩阵 n 个周期。下面说明上述特例需要 n 个周期的领先适用于任意的 $i (1 \leq i \leq n)$ 。当 A 开

始传的时刻落后 B 矩阵 n 个周期时， B_{xj} 传到 PE_j 时的周期为 $(x-1)n+2j-1$ ，对于任意的 $1 \leq i \leq n$ ， A_{ix} 到达 PE_j 的周期为 $n+(x-1)n+i+j-1$ ，比较并化简所需周期数得到结果为 $j < n+i$ 。由于 $j < n+i$ 显然成立，故 B 矩阵的传输要领先于 A 矩阵 n 个周期传输。对于第 2 个要求，本文从少到多依次增加寄存器数量，发现 2 个寄存器能够满足要求，下面进行说明。在 2 个寄存器的情况下，假设已经存了 B_{xj} 和 $B_{(x+1)j}$ ，现在 $B_{(x+2)j}$ 要取代 B_{xj} ， B_{xj} 能被取代的条件是 A_{nx} 已经通过 PE_j ，这说明 A_{nx} 到达 PE_j 的周期数小于 $B_{(x+2)j}$ 到达 PE_j 的周期数。

A_{nx} 到达 PE_j 的周期数为 $n+(x-1)n+i+j-1$ ， $B_{(x+2)j}$ 到达 PE_j 的周期数为 $(x+1)n+2j-1$ ，由于 $n+(x-1)n+i+j-1 < (x+1)n+2j-1$ 成立，所以需要另外 2 个寄存器存储输入矩阵 B 的元素值。

本文实验表明，CNN 卷积核矩阵中 0 值很少而特征图矩阵 0 值非常多，稀疏性由特征图矩阵决定。上述结构中矩阵 A 的值要传输整个线性脉动阵列，矩阵 B 可以不传完。根据最大化稀疏性的特点，特征图矩阵以 A 形式传输，卷积核矩阵以 B 矩阵形式传输。本文的单输出线性脉动阵列相对于文献[13-14]有如下 3 方面的改进（多输出线性脉动阵列没有第 3 方面，具体改动在第 4 节说明）。

1) 对于有很多 0 值的输入特征图 A 矩阵，在传送之前进行判定，为非 0 值时传入，当为 0 值时，该周期直接不传，相当于执行一个空操作，于是，就少 n 次（线性脉动阵列的 PE 数量）传输操作、 n 次乘操作和 n 次加操作。判断用到比较器，根据文献[11]，在 FPGA 中比较器可通过 LUT 构建，相比于需要利用 DSP48E 完成的乘加操作，增加的判断操作所需时间以及耗能很少。

2) 卷积核矩阵 B 在进行传送时不用传遍整个线性脉动阵列， B_{11} 传送到 PE_1 时被存储，继续往后传没有意义，因为后面的 PE 单元不再用 B_{11} 。为此，本文改进 B 矩阵的传输模式，当 B_{xj} 传输到 PE_j 时，进行类似“剪切”加“粘贴”的操作，把 B_{xj} 的值放进 PE_j 的寄存器后不再传送 B_{xj} 的值。对于该操作， B_{x1} 少传了 $n-1$ 次， B_{x2} 少传了 $n-2$ 次，依次类推一行共减少 $1+2+\dots+(n-1) = \frac{n^2(n-1)}{2}$ 次传输操作， B 矩阵

减少 $\frac{n^2(n-1)}{2}$ 次寄存器间的传输操作。

3) 对于输出矩阵, 为了实现最后输出能和乘法运算同时运行, 文献[13-14]采用 Cobuf 和 Cbuf 2 个 n byte (深度) SRAM Cache。本文只采用一个 n 深度的 SRAM Cache, 同时保证运算和传输不冲突。当然, 本文 PE 内 Cache 要能同时满足中间结果以及最终结果的存取需要 4 端口, FPGA 中没有直接的 4 端口 SRAM, 可以通过加控制器控制多个全双端口 BRAM 达到要求^[20]。下面结合图 9 说明一个 Cache 就能满足运算和传输不冲突的要求。

图 9 以 3×3 的矩阵乘法进行示例说明。“√”代表得出最后的计算结果且下一个周期进行传输, “←”代表传送方向, “有”代表该位置有值继而下一个周期往前传, “空”代表该位置已经把值传送出去。例如, 第 1 周期内完成了 C_{11} 的最终计算, 第 2 周期把 C_{11} 直接传送出去, 同时完成 C_{21} 和 C_{12} 的计算而不影响 C_{11} 的传输。第 3 周期把 C_{21} 直接传送出去, 同时把 PE2 中的 C_{12} 传输到 PE1 中, 此周期完成 3 个结果的最终计算。该模式下 PE 传输数据按照从上到下的顺序依次循环进行, 如第 4 周期时, PE₁ 中已经有 2 个待传的数据 (第 3 周期结束时 PE₁ 中的“有”和“√”), 但此时按照从上到下只传最下面的一个规则, 从得出第一个最终值开始, 往后的每个时刻都能往外输出矩阵 C 的一个值, 第 10 周期完成所有值的传输, 这期间计算可以和传输同时进行而不会有冲突。图 10 给出单输出线性脉动阵列完成 3×3 矩阵乘法。

4 单输出线性脉动阵列性能分析

本文所用的单输出线性脉动阵列和文献[11]的并行矩阵乘法器的对比从 3 个方面进行: 计算循环周期、存取操作类别和存取操作次数以及资源耗费情况。下面依次进行说明 (接下来分析针对没有 0 值的情况, 为 0 值时对应线性脉动阵列的第一个方面改进)。

4.1 计算循环周期

对于由 n 个 PE 完成的 $n \times n$ 的矩阵乘法, 单输出线性脉动阵列共需要 $2n^2 + 1$ 个周期完成所有操作 (从开始输入到输出所有数据)。

本文把所有操作分为 2 个阶段, 2 个阶段以 A_{1n} 开始传入分界点, A_{1n} 开始传入线性脉动阵列时已进行的周期数为 $n + (n-1)n + 1 + 1 - 1 = n^2 + 1$, 之后每个周期输出一个值到线性脉动阵列外部, 共需要 n^2 个周期, 总共所需周期数为 $2n^2 + 1$ 。

对于并行矩阵乘法器, 当有 n 个处理单元时, 由于每个周期从 A 输入端口只输入一个数据, 所以共需要 n^2 个周期完成所有操作。时间上来说, 单输出线性脉动阵列是并行矩阵乘法器的 2 倍。

4.2 存取操作类别和存取操作次数

单输出线性脉动阵列和并行矩阵乘法器在完成 $n \times n$ 的矩阵乘法时, 要从片外内存取数据到片内缓存以及反向的存操作, 这部分操作是一致的, 另外, 每个 PE 内部深度为 n 的 SRAM Cache 存取中间结果也是一致的。

对于单输出线性脉动阵列, 从 PE 外 Cache 输入数据时, 每个数据只输入一次, 即 $2n^2$ 次 Cache 到寄存器操作, 之后在线性脉动阵列 PE 的

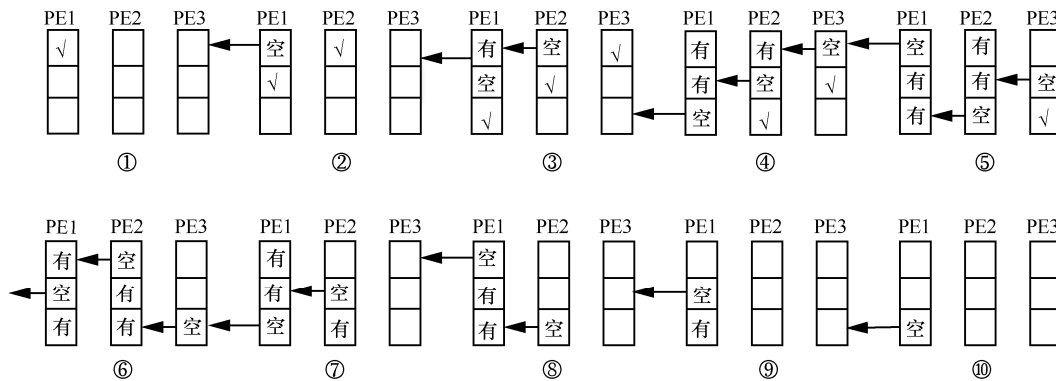


图9 单 Cache 传输情况

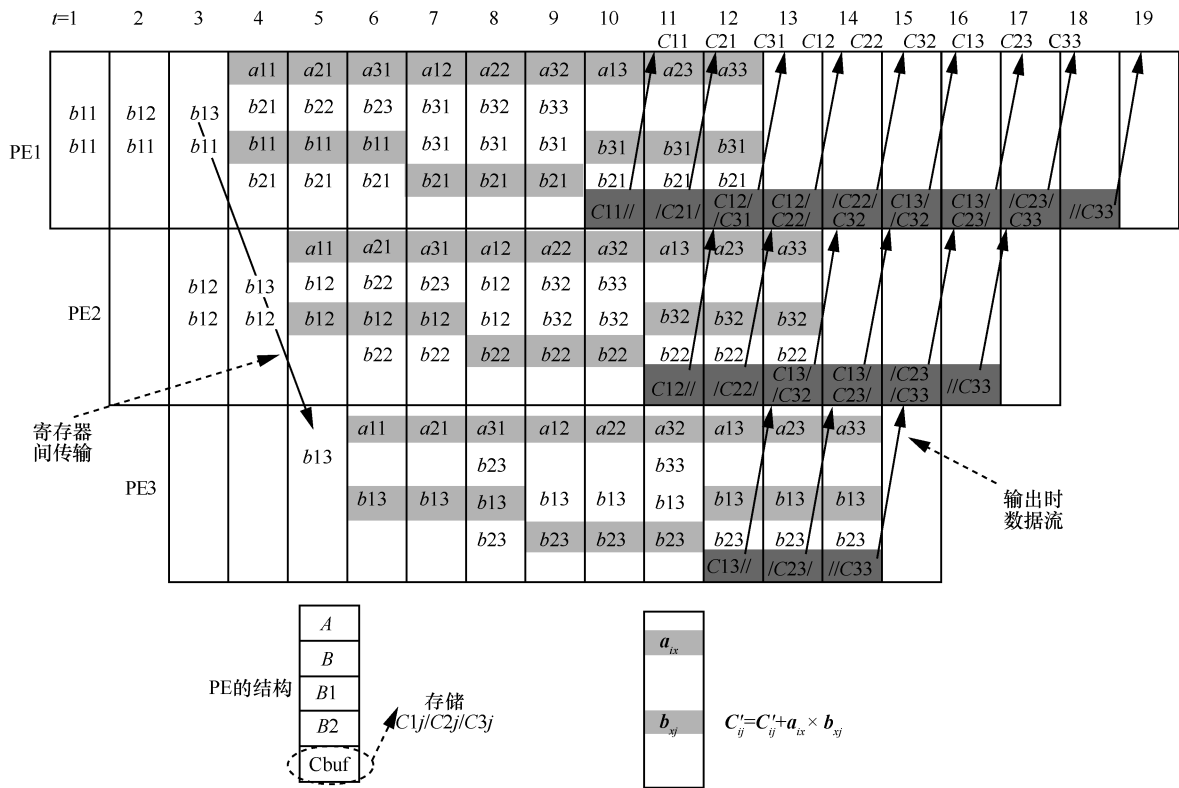


图 10 单输出线性脉动阵列实现矩阵乘法操作过程

内部寄存器之间传输， A 矩阵每个数据都要传遍所有 PE（即一个数据要在寄存器间传 $n-1$ 次），总计 $n^2(n-1)$ 次寄存器之间传输操作， B 矩阵数据由于“截取”操作，一行的各个数据所有传输的次数不一样，总计 $n(0+1+\dots+(n-1)) = \frac{n^2(n-1)}{2}$ 次寄存器间传输操作， B 矩阵加起来总计 $\frac{3n^2(n-1)}{2}$ 寄存器间传输操作。往外存时都是在 SRAM 中进行的，共 $n+2n+\dots+n^2 = \frac{n^2(n+1)}{2}$ 次寄存器间传输操作。

对于并行矩阵乘法器， $n \times n$ 的 A 矩阵每个数据每次都要读取 n 次（ n 个 PE 每次共用一个 A 中的值），即读取 PE 外 Cache 中数据存到 PE 内寄存器的操作总计 n^3 次。 A 矩阵一行循环完，就把 B 矩阵读取一遍（即 n^2 次读取操作），所以 B 矩阵共需要 $n \times n^2 = n^3$ 次读操作， A 、 B 加起来为 $2n^3$ 。从 PE 内深度为 n 的 SRAM Cache 存数据到外部 Cache 的次数为 n^2 ，对比情况如表 2 所示。

表 2 单输出线性脉动阵列和并行矩阵乘法器存取操作类别和存取操作次数对比

操作类别	存取操作次数	
	单输出线性脉动阵列	并行矩阵乘法器
片外→片内及片内→片外	一致	一致
PE 外 Cache→PE 内寄存器	$2n^2$	$2n^3$
PE 内部寄存器→寄存器	$\frac{3n^2(n-1)}{2}$	—
存取中间结果 Cache 操作	一致	一致
输出时 Cache 操作	$\frac{n^2(n-1)}{2}$	n^2

从上述对比可以看出，二者的数据搬移量相等。从能量方面看，PE 外 Cache 到 PE 内寄存器传输操作以及输出时的 Cache 操作可以视为从 SRAM 读取数据的能量消耗，可以进行合并。差别如下：单输出线性脉动阵列独有 $\frac{3n^2(n-1)}{2}$ 次寄存器到寄存器的传输操作，而并行矩阵乘法器多 $\frac{3n^2(n-1)}{2}$ 次 SRAM 读取操作。

从表 1 看出，寄存器读和 SRAM 读的能量消耗是 1 和 10 的关系，即并行矩阵乘法器多

$$(5-1)\frac{3n^2(n-1)}{2} = 6n^2(n-1)\text{pJ 能量。}$$

4.3 资源耗费情况

改进的线性脉动阵列 B 输入需要额外的 2 个寄存器, 相比并行矩阵乘法器多消耗 $2n$ 个寄存器 (都是几个 PE 情况下)。如表 3 所示, 并行矩阵乘法器的 FF (寄存器) 消耗量是比较少的, 不是主要的资源限制因素, 影响可以不予考虑。

表 3 并行矩阵乘法器用于 Alexnet 卷积加速时的 FPGA 内部资源使用情况

资源类别	总使用量	可利用总量	利用率
BRAM_18K	1 458	2 060	70.78%
DSP48E	1 792	2 800	64.00%
FF	1 792	607 200	28.12%
LUT	142 304	303 600	46.87%

从 I/O 端口来看, 单输出线性脉动阵列只需 3 个端口, 而并行矩阵乘法器需要 $n + n + n = 3n$ 个端口, 对于 n 较大的情况, 端口消耗以及带宽要求差异非常明显。从资源消耗来看, 单输出线性脉动阵列虽然使用较多的 FF 资源, 但却很大程度上减少了 I/O 端口量及带宽要求。

4.4 与并行矩阵乘法器对比

单输出线性脉动阵列和并行矩阵乘法器各有优缺点, 具体的对比分析如表 4 所示。从表 4 可以看出, 虽然时间耗费增多, 但本文单输出的线性脉动阵列很好地弥补了传统的并行矩阵乘法器带宽要求太大的问题。

表 4 单输出线性脉动阵列和并行矩阵乘法器对比

	单输出线性脉动阵列	并行矩阵乘法器
周期	$2n^2+1$	n^2
存取操作差异次数	$\frac{3n^2(n-1)}{2}$ (寄存器读)	$\frac{3n^2(n-1)}{2}$ (SRAM 读)
寄存器消耗	多 $2n$	—
I/O 端口及带宽	3	$3n$

5 多输出线性脉动阵列性能分析

当结合具体 CNN 模型 (如 Alexnet) 进行分析时, 卷积核矩阵以及输入特征图矩阵太大, 需要进行分块操作。文献[11]的单列并行矩阵乘法器用于 Alexnet 卷积加速时采用了 448(14×32)个

PE, 为更好地进行理论上的对比, 本文也采用 14 个并行的线性收缩阵列, 每个有 32 个 PE。为了有直观上的认识, 把 Alexnet 各卷积层矩阵分块为大小为 32×32 的小矩阵后的情况列举出来以方便后面的分析, 具体如表 5 所示。

表 5 Alexnet 各卷积层矩阵分块

卷积层	原始大小	分块
卷积层 1	(3 025×363)×(363×96)	(95×12)×(12×3)
卷积层 2	(729×1200)×(1200×256)	(23×38)×(38×8)
卷积层 3	(169×2 304)×(2 304×384)	(6×72)×(72×12)
卷积层 4	(169×1 728)×(1 728×384)	(6×54)×(54×12)
卷积层 5	(169×1 728)×(1 728×256)	(6×54)×(54×8)

分块后一个很重要的想法是进一步复用和并行流水线。14 个并行的线性脉动阵列可以同时并行进行矩阵乘法计算, 分块矩阵乘法有以下 2 种 (如图 11 所示): 第一种是为了在乘法完成后方便进行相加; 第二种是为了实现对分块矩阵的复用, 设置多级缓存可以进一步减少数据搬移量。

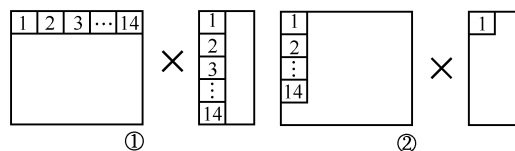


图 11 线性脉动阵列 2 种分块矩阵乘法方式

线性脉动阵列计算只是占了整个过程的一部分, 如图 10 中前 3 后 5 个周期都没有计算操作, 单纯的传输时间占用不少。除了复用, 还可以采用流水线设计。 A 起始传输是落后 B 矩阵 n 个周期的, 按常理可以把 A 、 B 矩阵传输时无缝结合, 第一个分块矩阵乘法在传最后一列 A 矩阵元素时也传输下一个分块矩阵乘法的 B 矩阵的第一行, 这样除了最开始和最后的分块矩阵传输时是单线输入外 (最开始的 n 周期只有 B 输入, 最后的 n 周期只有 A 输入), 其他时刻都是双线输入。但上述想法实施起来会出现矛盾, 因为最后输出时是 n^2 个周期, 大于 n 个周期的错位, 直接无缝传输时前一个分块矩阵乘法结果还没有输出到 PE 外第二个分块矩阵乘法的中间结果就要存到 PE 内 Cache, 这样就会有冲突。为了进一步利用流水线, 可以把最终结果尽快输出到 PE 外的 Cache, 即用 n 输出端口加快数据输出到 PE 外

Cache, 这样就能做到无缝输入, 本文把这种对单输出线性脉动阵列和并行矩阵乘法器的整合改进称为多输出 (“多” 体现在输出端口) 线性脉动阵列。这种多输出线性脉动阵列最后完成所有分块矩阵乘法的周期变为 $xn^2 + 2n$ (x 指单个 PE 所进行的分块矩阵乘法的个数), 而并行矩阵乘法器为 xn^2 , 当以表 5 的 Alexnet 第一个卷积层分块情况来具体比较时 (此时 x 的值为 $\left\lceil 92 \times 12 \times \frac{3}{14} \right\rceil = 245$, n 的值为 32), 两者的倍数关系为 1.0003 倍, 相比之下几乎没有差别。

另外, 采用这种方式的存取操作类别和存取操作次数相比单输出线性脉动阵列只有输出时的 Cache 操作有变化, 变成和并行矩阵乘法器一样的 n^2 。这种多输出线性脉动阵列和并行矩阵乘法器完成一个分块矩阵乘法的对比结果如表 6 所示, 其中多输出线性脉动阵列的周期是平均值, 参数 x 指单个 PE 所要完成的分块矩阵乘法的个数。三者的特点总结如表 7 所示。

表 6 多输出线性脉动阵列和并行矩阵乘法器存取操作类别和存取操作次数对比

	多输出线性脉动阵列	并行矩阵乘法器
周期	$xn^2 + \frac{2n}{x}$	n^2
存取操作差异次数	$\frac{3n^2(n-1)}{2}$ (寄存器读)	$2n^2(n-1)$ (SRAM 读)
寄存器消耗	多 $2n$	—
I/O 端口及带宽	$2+n$	$3n$

表 7 改进的线性脉动阵列和并行矩阵乘法器特点

改进方案	特点
文献[11]并行矩阵乘法器	带宽要求最大但周期最少, 耗能最多
单输出线性脉动阵列	带宽要求最小但周期最多, 耗能次之
多输出线性脉动阵列	带宽、周期处于前两者之间, 耗能最少

相比单输出线性脉动阵列, 多输出线性脉动阵列的最大优点是周期数大幅减少。相比并行矩阵乘法器, 该方式仍然具有带宽的优势。具体应用场景方面, 由于线性脉动阵列能利用输入中全部稀疏性 (A 、 B 中任意一者的稀疏性都能利用), 对于像文献[19]中剪枝技术应用后带来的卷积核稀疏性, 此时线性脉动阵列的优势明显大于并行矩阵乘法器 (并行矩阵乘法器只能利用某一方输

入中稀疏性)。

6 结束语

本文针对 CNN 计算稀疏性的特点以及文献[11]所述的传统并行矩阵乘法器用于 CNN 加速的局限性, 对文献[13-14]的线性脉动阵列提出了 2 种改进——单输出以及多输出线性脉动阵列, 用于 CNN 嵌入式端的加速。文献[11]提出的并行矩阵乘法器直接作用于 CNN 加速时所需带宽太大, 而线性脉动阵列正好能弥补该缺陷。单输出线性脉动阵列和并行矩阵乘法器对比有耗能以及 I/O 端口少的优势, 但计算周期却是 2 倍的关系, 所以本文进一步改进线性脉动阵列, 提出了多输出线性脉动阵列。对比来看, 多输出方式的周期及 I/O 需求介于两者之间, 很好地解决了前两者的突出缺陷问题, 可行性最高。本文所进行的对比分析只是理论上说明, 如何最佳组合以及最充分利用资源并实现是下一步的工作。

参考文献:

- [1] HAN S, MAO H, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding[J]. Fiber, 2015, 56(4): 3-7.
- [2] QIU J, WANG J, YAO S, et al. Going deeper with embedded FPGA platform for convolutional neural network[C]// International Symposium on Field-Programmable Gate Arrays. 2016: 26-35.
- [3] SABOUR S, FROSST N, HINTON G E. Dynamic routing between capsules[C]//Annual Conference on Neural Information Processing Systems. 2017.
- [4] HAN S, LIU X, MAO H, et al. EIE: efficient inference engine on compressed deep neural network[J]. ACM Sigarch Computer Architecture News, 2016, 44(3): 243-254.
- [5] CHEN W, WILSON J, TYREE S, et al. Compressing neural networks with the hashing trick[C]//International Conference on Machine Learning. 2015: 2285-2294.
- [6] MA Y, CAO Y, VRUDHULA S, et al. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks[C]//International Symposium on Field-Programmable Gate Arrays. 2017:45-54.
- [7] LI N, TAKAKI S, TOMIOKAY Y, et al. A multistage dataflow implementation of a deep convolutional neural network based on FPGA for high-speed object recognition[C]//2016 IEEE Southwest Symposium on Image Analysis and Interpretation. 2016: 165-168.
- [8] SUDA N, CHANDRA V, DASIK A G, et al. Throughput-optimized openCL-based FPGA accelerator for large-scale convolutional neural networks[C]// International Symposium on Field-Programmable Gate Arrays. 2016: 16-25.
- [9] XIAO Q, LIANG Y, LU L, et al. Exploring heterogeneous algo-

- rithms for accelerating deep convolutional neural networks on FPGAs[C]//The 54th Annual Design Automation Conference. 2017: 62-67.
- [10] CHEN Y H, KRISHNA T, EMER J S, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.
- [11] 刘勤让, 刘崇阳. 利用参数稀疏性的卷积神经网络计算优化及其 FPGA 加速器设计[J]. 电子与信息学报, 2018, 40(6): 1368-1374.
LIU Q R, LIU C Y. Calculation optimization for convolutional neural networks and FPGA-based accelerator design using the parameters sparsity[J]. JEIT, 2018, 40(6): 1368-1374.
- [12] LIU X, HAN S, MAO H, et al. Efficient sparse-winograd convolutional neural networks[C]//International Conference on Learning Representations. 2017.
- [13] JANG J W, CHOI S B, PRASANNA V K. Energy-and time-efficient matrix multiplication on FPGAs[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2005, 13(11): 1305-1319.
- [14] MATAM K K, LE H, PRASANNA V K. Energy efficient architecture for matrix multiplication on FPGAs[C]//International Conference on Field Programmable Logic and Applications. 2013:1-4.
- [15] JIA Y, SHELHAMER E, DONAHUE J, et al. Caffe: convolutional architecture for fast feature embedding[C]//The 22nd ACM International Conference on Multimedia. 2014: 675-678.
- [16] JIA Y Q. Optimzing conv in caffe[R].
- [17] MOONS B, DE BRABANDERE B, VAN GOOL L, et al. Energy-efficient convnets through approximate computing[C]// Applications of Computer Vision. 2016: 1-8.
- [18] 田翔, 周凡, 陈耀武, 等. 基于 FPGA 的实时双精度浮点矩阵乘法器设计[J]. 浙江大学学报(工学版), 2008, 42(9):1611-1615.
TIAN X, ZHOU F, CHEN Y W, et al. Design of field programmable gate array based real-time double-precision floating-point matrix multiplier[J]. Journal of Zhejiang University (Engineering Science), 2008, 42(9):1611-1615.
- [19] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network[C]//Annual Conference on Neural Information Processing Systems. 2015: 1135-1143.
- [20] LAI B C C, LIN J L. Efficient designs of multi-ported memory on FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 25(1): 139-150.
- [21] CHEN J, LI J. The research of peer-to-peer network security[C]//The International Conference on Information Computing and Automation. 2015: 590-592.

[作者简介]



刘勤让(1975-), 男, 河南睢县人, 国家数字交换系统工程技术研究中心研究员, 主要研究方向为宽带信息网络、片上网络设计。

刘崇阳(1994-), 男, 湖北宜昌人, 国家数字交换系统工程技术研究中心硕士生, 主要研究方向为人工智能、深度学习。

周俊(1979-), 男, 湖北黄冈人, 国家数字交换系统工程技术研究中心讲师, 主要研究方向为芯片设计、宽带信息处理。

王孝龙(1993-), 男, 河南民权人, 国家数字交换系统工程技术研究中心硕士生, 主要研究方向为宽带信息网络、协议解析。