doi:10.3772/j.issn.1002-0470.2019.03.003

稀疏神经网络加速器设计①

周聖元②*,**,*** 杜子东* 陈云霁*,**

(*中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

(**中国科学院大学 北京 100049)

(***上海寒武纪信息科技有限公司 上海 201306)

摘 要 针对日益增长的神经网络规模和不断变化的神经网络模型结构,提出了一款新型的稀疏神经网络加速器架构。该架构能够有效利用稀疏神经网络中的权值稀疏性和神经元稀疏性,进一步提升加速器处理神经网络模型时的运算速度。同时,该架构能够支持逐元素乘法/加法等运算,从而进一步提高加速器的灵活性,高效支持并加速 Resnet 等新型的神经网络结构。实验结果显示,基于5 个具有代表性神经网络模型,该架构相比于现有的先进的稀疏神经网络加速器有平均为 2.57 倍的加速比,同时针对 Resnet-18 和 Resnet-50 的 BN 层分别平均有 4.40 倍和 4.57 倍的加速比。

关键词 神经网络,稀疏神经网络,加速器

0 引 言

近些年来,神经网络算法迅速发展,已经广泛且深入地被应用到许多不同的领域,包括图像分类[1]、视频描述生成[2]、语音识别[3]、机器翻译[4]等等。在发展过程中很多具有代表性的神经网络模型不断被提出,如 Alexnet[5],VGG[6],Resnet[7]等,其特性也随之不断地发生着变化。

一方面,稀疏神经网络的出现使得神经网络的数据特性发生了变化。随着神经网络技术的不断发展,神经网络规模不断扩增,使得计算量和访存量急剧增加。因此,很多研究人员开始考虑将稀疏技术[8,9]应用到神经网络模型中,即将大量的神经元和/或权值置零,从而能够有效降低运算中有效数据数量和模型中参数数量。另外,修正线性函数(rectified linear unit, ReLU)[10]逐渐成为了应用最为广泛的激活函数,由于该激活函数能够将值小于 0 的神经元置零,于是,进一步提高了置零神经元的数量。大量的零值虽然可以降低运算量,但是也提高了数据处理的复杂程度,因为数据从原来可以规整的存储、读取和运算变得极为不规整,所以如何在加速器设计中能够有效利用稀疏神经网络的数据特性以降低运算量是一个值得研究的问题。

另一方面,随着神经网络模型的迅速发展,神经网络模型中运算特性也在逐渐发生变化,以 1998 年提出的 Lenet[11] 和 2015 年提出的 Resnet 为例。Lenet 中,包含经典的卷积层、池化层和全连接层,且这三层的占比基本相同;而在 Resnet 网络模型中,池化层和全连接层的比重大大降低,留有大量的卷积层,并且还包含大量的 BatchNorm 层和 Scale 层,其运算模式和经典的卷积层、全连接层并不相同,主要是逐元素进行算术运算。因此,随着不同的网络层被加入其中,神经网络的运算多样性成为了不容

① 国家重点研发计划(2017YFA0700900, 2017YFA0700901, 2017YFA0700902, 2017YFB1003101),国家自然科学基金(61432016, 61532016, 61672491, 61602441, 61602446, 61732002, 61702478),北京市自然科学基金(JQ18013),973 计划(2015CB358800),中国科学院科技成果转移转化重点专项(KFJ-HGZX-013),"核心电子器件、高端通用芯片及基础软件产品"科技重大专项(2018ZX01031102)和中国科学院战略性先导科技专项(XDB32050200, XDC01020000)资助项目。

② 女,1991 年生,博士生;研究方向:机器学习加速器,神经网络加速器;E-mail: zhousy@ict.ac.cn

(收稿日期:2018-07-10)

忽视的问题。

面对上述神经网络的发展趋势,现有的通用处理器 CPU 和 GPU 显然不能够提供有针对性的改变,从而提高性能并降低功耗。本文对神经网络算法的数据稀疏性和运算多样性进行了分析,并根据分析结果,设计了一款新型的神经网络加速器架构。实验表明,本文提出来的新型架构能有效应对上述神经网络的两个特性,相比于现有的先进的稀疏神经网络加速器能够明显地提升性能。

本文的主要贡献如下:

(1) 提出了一款新型的稀疏神经网络加速器,该加速器能够有效支持多种神经网络模型的推测运算,能够比现有的先进的加速器具有更高的性能。

(2) 从数据特性方面对神经网络模型的不同的数据稀疏性进行了统计和分析,有针对性地设计了加速器的选数部分,从而能够保证运算部分的运算有效性。

(3) 从运算特性方面对不同的神经网络层的占比和特点进行了分析,从而提出了新型的运算单元结构,从而能够有效适应神经网络发展过程中带来的运算多样性。

1 相关工作

随着神经网络的迅速发展,急剧增大的数据量和运算量给计算机系统带来了很大挑战。虽然通用处理器 CPU 和 GPU 能够支持不断变化的运算模型,但是也正是因为其为了保证通用性,无法提供较高的性能和较低的功耗。于是神经网络加速器应运而生,并得到了迅速发展。在 2014 年,Chen 等人[12]提出了 DianNao,能够仅在 3.02 mm^2 的面积内达到主流 CPU 性能的 117.87 倍,同时其功耗开销仅为 CPU 的 4.74%。而后,该工作中的神经网络加速器的出色性能和极低功耗,引起了大量学者的关注和深入研究。DaDianNao[13]采用片上 eDRAM 作为存储来降低访存。Du 等人[14]提出了可以集成到嵌入式设备的 ShiDianNao,从而实现端到端的神经网络应用解决方案。PRIME[15]设计了一种专门针对神经网络的 Re-RAM。ISAAC[16]采用忆阻器作为存储单元,同时采用模拟电路作为运算单元。但是这些加速器架构仅能够支持稠密的神经网络模型的相关运算。

随着 2015 年稀疏神经网络的提出[17],又出现了很多针对稀疏神经网络的加速器架构。Han 等人[18]设计的 EIE 能够利用稀疏神经网络模型中的权值稀疏性和神经元稀疏性,该加速器采用了多处理单元(processing element, PE)的架构,但是只能够有效加速全连接层的运算。Cnvlutin[19]同样可以同时利用权值稀疏性和神经元稀疏性,但是只针对卷积层进行了加速和讨论。Zhang 等人[20]提出的 Cambricon-X 能够有效加速卷积层和全连接层,但是仅能够利用权值的稀疏性。SCNN[21]提出了一种新型的数据流,能够兼顾卷积层和全连接层的权值稀疏性和神经元稀疏性的运算,但是其功耗和面积开销都相对较大。

上述方法虽然都取得了一定的性能功耗,但是无法在一个较小的面积和功耗开销下兼顾多种数据稀疏性的同时有效支持多种网络层的运算。本文提出的稀疏神经网络加速器架构能够有效应对神经网络模型的多种数据稀疏性,能够有效支持新型神经网络模型的结构和网络层的多样性,进一步增加运算效率。

2 神经网络的数据特性

本节主要介绍随着神经网络的数据特性,即数据稀疏性。2.1 节简要介绍数据稀疏特性的种类,2.2 节进一步分析神经元稀疏性。

2.1 稀疏性的种类

权值稀疏性 随着神经网络剪枝技术的出现和发展,稀疏神经网络的稀疏性逐渐获得了研究人员的关注。在韩松等人[17]的研究中,通过剪枝等操作之后,Alexnet、VGG16 等网络模型的权值稀疏度(指非零数据数量占数据总数量的比率)能够高达 11%和 7.5%,同时对网络的运算精度不产生影响。显然,神经网络权值稀疏性能够大大降低神经网络数据的数据量和运算量。以 Cambricon-X 为例,该架构由于有效利用权值稀疏性这一特性,相比于同为类向量点积结构的 DianNao 获得了7.23倍的加速

比。然而,Cambricon-X 仅利用了神经网络的权值稀疏性,而没有考虑到神经元的稀疏性。

神经元稀疏性 随着神经网络的发展,越来越多的神经网络模型的激活层采用 ReLU 函数作为激活函数,即 $f(x) = \max(0, x)$。显然,该函数将所有的负数值置零,从而也产生了大量零值的神经元,大大提高了神经元的稀疏性。由于零值与另一数据进行加法时,结果和另一数据数值相等;零值与另一数据进行乘法,结果为 0,因此零值参与的运算都可以认为是无效运算,可以被跳过。据统计,在运算过程中,卷积层平均有 44% 的运算操作为无效操作[21],因为至少有其中一个操作数为 0。

2.2　神经元稀疏性

神经元稀疏性主要是激活函数 ReLU 在每次运算后对数据数值进行判断而带来的,或者根据设定阈值对数据强制设为 0[19],所以和权值稀疏性不同,神经元的稀疏数据无法通过和权值剪枝等类似的操作进行预先固定,如图 1 所示,其稀疏度会因为输入数据不同、网络结构不同;如图 2 所示,以 Alexnet 网络模型为例,相同输入数据,相同网络结构,网络层不同而稀疏度也会不同。所以,对于神经元稀疏,必须能够在运算过程中进行筛选,得到有效的运算数据。

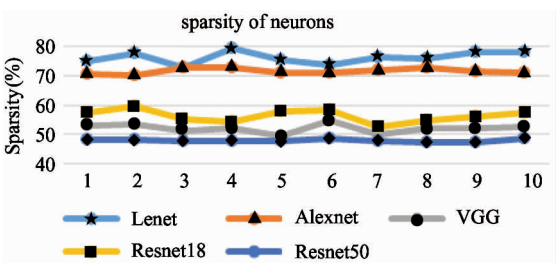


图 1　不同输入数据的神经元稀疏度

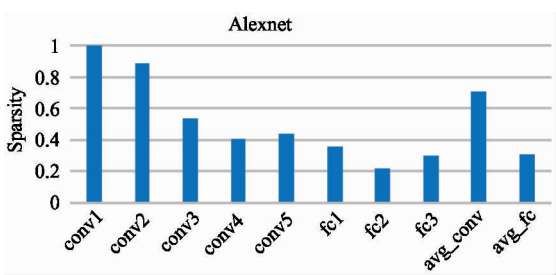


图 2　Alexnet 模型中不同层的神经元稀疏度

3　神经网络的运算特性

本节主要介绍神经网络的运算特性,其中,3.1 节给出了各层运算时间的 Caffe[22] 架构下测得的运算时间,3.2 节进一步分析了各层的运算特性。

3.1　各层运算时间占比

随着神经网络的不断发展,新的网络结构层出不穷,如比较具有代表性的模型:Alexnet、VGG、Resnet 等。其中,除了原本常见的卷积层、全连接层等,也出现了很多不同种类的网络层以及组合方式。以 Resnet 为例,其利用 Caffe 的 time 功能分别测试了多种神经网络模型(包括 Lenet, Alexnet, VGG16, Resne-18 和 Resnet-50 等)在 GPU(Tesla K40c)上的推测阶段的运行时间。如图 3 所示,尽管经典层(卷积层 + 全连接层)依然占据了大部分的运行时间,但是其所占比例大大降低了。对 Lenet、Alexnet 和 VGG16 而言,这两种层共占运算时间的81.75%、90.15% 和 94.42%。而对于 Resnet-18 和 Resnet-50 的全连接层的比例大大降低至 0.34%,而卷积层所占时间比例也明显地降低,分别只占有 66.22% 和 58.00%。

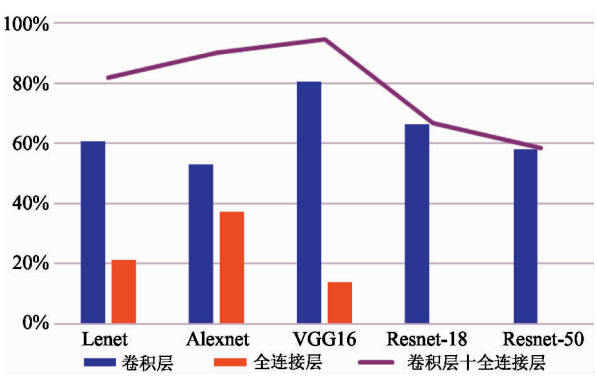


图 3　5 种网络模型中卷积层和全连接层在 GPU 的计算时间比

进一步地,对 Resnet-18 和 Resnet-50 网络模型中各网络层在 CPU(Intel(R) Xeon(R) CPU E5-2620 v2)和 GPU(Tesla K40c)上的推测阶段的运行时间进行分析,如图 4 所示。其中,BatchNorm 层和 Scale 层所占比例明显,分别占 Resnet-18 网络模型运算时间的 18.7% 和 6.4%,占 Resnet-50 网络模型

运算时间的22.97%和8.63%。在CPU上,这两类层的运行时间也分别占Resnet-18和Resnet-50网络模型的14.94%和13.33%。所以,对于一款神经网络加速器而言,仅仅考虑卷积层和全连接层是不够的,BatchNorm层和Scale层也是不容忽视的。另外,选用不同的Batch进行测试,各层运算时间所占比例变化不大。

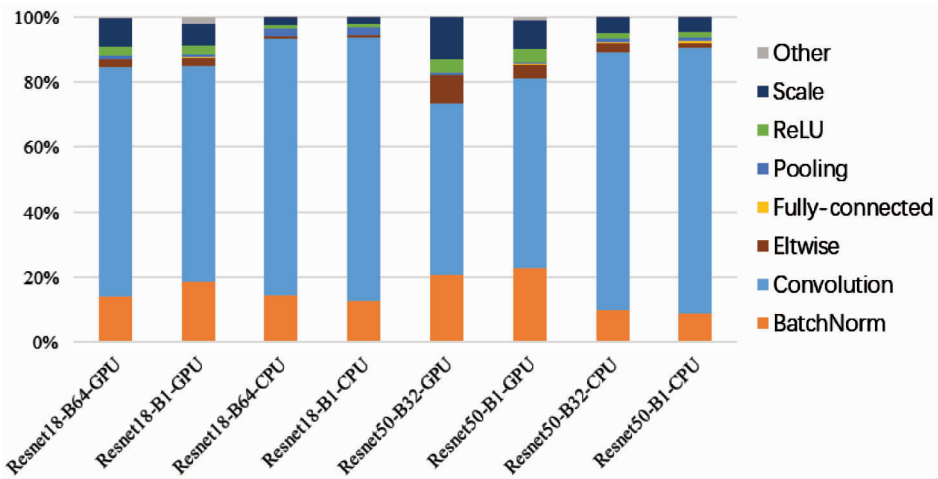


图4　Resnet-18网络和Resnet-50网络中各网络层计算时间比

3.2　BN层和Scale层的特点

神经网络经典层的运算以卷积层为例,如式(1)所示,输出神经元的值通过神经元数据和权值数据相乘而后累加,并激活得到。

$$Out = f(\sum(\boldsymbol{input} \cdot \boldsymbol{weight})) \tag{1}$$

显然,采用类向量点积的运算模式,即将输入神经元和权值分别看作两个向量,对应元素进行相乘之后进行累加,能够有效加速其运算,并且该类结构已被证明能够获得较好的性能[12]。

BN层[23]用于对数据进行归一化操作,其主要步骤为求得batch内数据的平均值*mean*和标准差*std*,而后对数据进行归一化,即$x' = (x - mean)/std$。Scale层是通过缩放因子a和平移因子b对数据进行变换得到$y = ax' + b$。显然,这两个运算层和卷积层的运算特点不同,它们都是逐元素进行算数运算操作。由于元素间数据并没有相关性,理论上可以通过同时完成多组元素的运算来进一步提高运算效率。但是,显然地,当把该过程直接映射到已有的类向量点积的结构中时,该运算过程依然是个串行的过程,即一次只能完成一组元素的乘和/或加操作,无法对其进行加速,并且硬件利用率很低。

4　加速器架构

基于上述分析,本文提出了一款新型的稀疏神经网络加速器架构,该架构能够有效利用神经网络数据和权值数据的稀疏,高效完成常见神经网络的运算。

4.1　整体结构

加速器的整体架构基于DianNao的基本架构,即主要包括以下几个部分:控制处理器(control processor, CP)、2个片上缓存(shared buffer, SB,分别用SBin和SBout标记)、直接存储访问模块(direct memory access, DMA)和1个由多个处理单元(processing element, PE)组成的功能单元(functional unit, FU),如图5所示。为了避免线路拥堵,各处理单元利用胖树(FatTree)进行连接。

每个处理单元中,包含1个运算单元(computational unit, CU)、1个数据控制器(data controller, DC)和1个私有的片上缓存(private buffer, PB)。

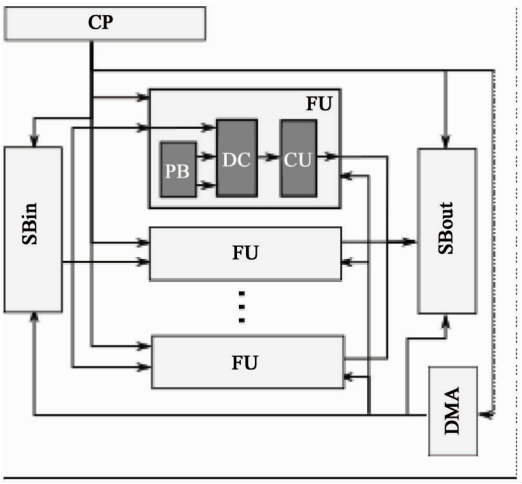


图 5　本加速器的整体结构图

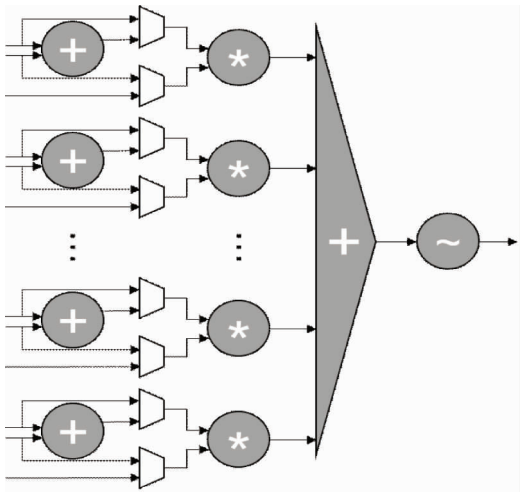


图 6　运算单元的结构图

4.2　运算单元

运算单元采用 4 级流水的架构，如图 6 所示，包括加法、乘法、加法树和非线性运算等，分别用于同时完成多组数据的逐元素加减法、逐元素乘法、加和和激活等运算。运算过程中，可以根据指令指定选用其中的阶段进行运算。如卷积、全连接层，输入数据直接输入到乘法器，利用乘法器、加法树和非线性运算阶段完成所需要的运算操作；在 BN 层，利用加法阶段完成神经元和平均值的逐元素相减，而后将运算结果传递给下一级乘法器和一个给定常数完成逐元素乘积操作，得到运算结果；在 Scale 层，输入数据跳过加法器阶段直接在乘法器阶段将输入的数据和一个给定常数相乘，完成输入数据的缩放操作。

具体来说，第 1 级流水级中，包括 16 个加法器，能够同时完成 16 对元素的两两相加或两两相减运算。第 2 级流水中，包括 16 个乘法器，同时完成 16 对元素的两两相乘的运算，其中，一个元素来自于缓存中，另一个元素可以来自于缓存中，也可以为前一级的加法器的运算结果。第 3 级为 16-输入的加法树，将乘法器的乘积进行加和。第 4 级为轻量级的 ALU，用于完成激活等非线性运算。其中，任意一级的结果均可作为运算单元的最终结果输出。

其中，由于神经网络具有一定的容错性，所以为了进一步减少功耗和面积，运算单元采用 16 位定点的格式[24]进行运算。

4.3　数据控制器

数据控制器用于对输入的数据进行筛选，仅把有效运算数据传递给后面的运算单元。具体过程如图 7所示，神经元数据从 SBin 中传递给 DC，稀疏权值数据和标记信息从 PB 中传送到 DC 中。首先，将神经元数据和已知的阈值（th）进行比较，筛选出有效的神经元，用神经元标记信息进行标记。该标记信息用 0 和 1 表明该位置的神经元数据是否有效，有效为1，无效为0。然后，将神经元标记信息和同

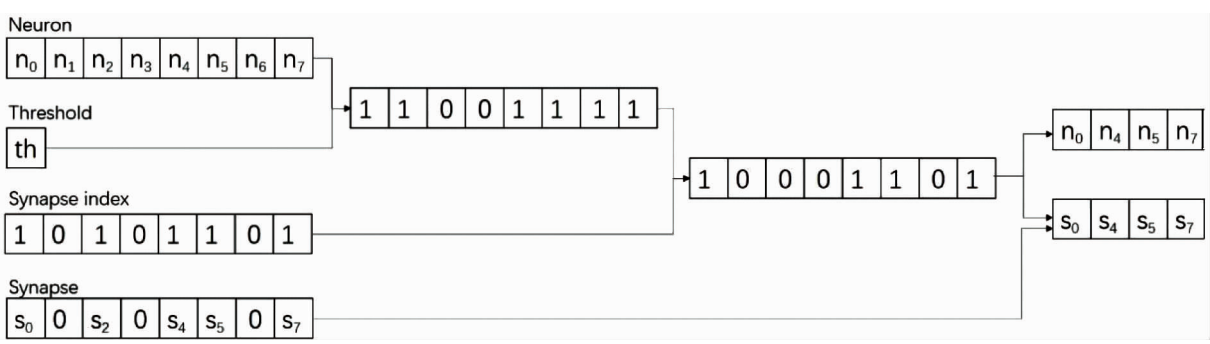


图 7　数据控制单元的功能示意图

样采用0和1表示方式的权值标记信息进行与操作,生成的同样用0和1进行标记的有效运算标记信息。最后,通过有效运算标记信息索引得到运算有效的神经元数据和权值数据,送入后面的运算单元进行运算。

虽然在单次筛选过程中,步长索引的方式比直接索引的方式略有优势[20],但是由于这里需要多次对数据进行比较和筛选,采用直接索引的方式能够更直接,减少索引转换的过程。

其中,将阈值(th)设定为0,即可将不为0的神经元作为有效神经元,得到有效神经元标记信息。如果在测试过程中,精确度要求不那么高,可以将阈值设为其他数值,从而进一步减少有效神经元的数量,降低运算量。而后通过有效神经元的标记信息和权值标记信息得到有效运算的标记信息。根据有效运算的标记信息,筛选出有效的待运算的神经元数据和权值数据。

5 映射方式

在本节中,将详细介绍神经网络层在本神经网络架构中的映射方式。

5.1 经典层

卷积层、全连接层等经典的神经网络层,采用和Cambricon-X中的相同的调度规则,数据从片上缓存中读取出来,经过各PE中的数据控制模块筛选出有效数据后,送入运算单元,运算单元根据控制单元,选择相应需要的流水级完成运算。其中,为了降低片上片下的数据访存,最大化地复用片上神经元数据。具体而言,当输入神经元和权值规模均大于加速器的规模时,将一部分神经元数据和权值数据载入到片上后进行运算。运算完毕后,替换片上的权值数据,即载入新的权值数据,而后和原来片上的神经元数据进行运算。依次类推,直到片上的神经元数据全部计算完毕,再对其进行替换。

5.2 BatchNorm层和Scale层

BN层用于对输入数据进行归一化处理,即 $x' = (x - mean)/std$。在推测阶段,平均值 $mean$ 和标准差 std 均为已知常数,故从PB中读取神经元数据 x,从SB中读取给定平均值mean和标准差的倒数 $1/std$,分别送入加法器完成相减操作和与前一拍的差值进行相乘操作。

Scale层是通过缩放因子 a 和平移因子 b 对数据进行变换得到 $y = ax' + b$。由于 a 和 b 均为已知常数,故可以经过预处理,将运算进行等式变换为 $y = a(x' + b/a)$。而后,类似的,从PB中读取神经元数据 x,从SB中读取参数 b/a 和 a,分别送入加法器和乘法器进行运算。

当BN层和Scale层连用的时候,由于在推测阶段,BN层的数据平均值和标准差为给定常数,故该过程中的参数均为常数,从而通过等式变换,将运算过程变换为 $y = (a/std) \cdot (x + (std \cdot b/a - mean))$,进一步简化了运算过程。

假定每个运算单元中可同时完成 n 对运算的逐元素运算,那么,每个运算单元中从PB中读取 n 个神经元数据 $x_1, x_2, \cdots, x_n$,从SB中读取预处理后的常数 $(std \cdot b/a - mean)$ 和 (a/std),分别送入加法器和乘法器。于是,在加法器阶段同时进行 n 个运算的加法运算,即得到 $(x_1 + (std \cdot b/a - mean))$, $\cdots$, $(x_n + (std \cdot b/a - mean))$,在乘法器阶段同时进行前一级的 n 个结果与 (a/std) 的乘法运算。

6 实验与结果

本节将介绍本文的实验环境和实验数据,并将分别从加速器的性能和功耗两个方面和Cambricon-X的进行了对比。

6.1 Benchmark

这里采用了5种具有代表性的神经网络模型,包括LeNet, Alexnet, VGG16, Resnet-18和Resnet-50。各网络模型中的平均的权值稀疏度和神经元稀疏度如表1所示,同时还列出了卷积层(CNV)和全连接层(FC)各自的稀疏度,以进一步说明本加速器能够有效利用神经网络的稀疏性进行加速。

6.2 数据集

本实验采用了Mnist和ImageNet两个数据集。这两个数据集均被广泛应用于神经网络算法的研究中。

表 1 各 Benchmark 的权值稀疏度和神经元稀疏度

Benchmark	Synapse			Neuron		
	Total(%)	CNV(%)	FC(%)	Total(%)	CNV(%)	FC(%)
Lenet	8.43	13.06	8.14	76.26	76.11	81.76
Alexnet	9.83	36.78	8.76	71.17	71.32	31.34
VGG-16	10.59	35.18	7.66	52.22	52.23	29.08
Resnet-18	29.39	29.13	35.00	56.40	56.40	98.91
Resnet-50	44.90	46.20	30.00	47.93	47.93	100.00

Mnist 是一个黑白手写数字图库,每张图片包含 28 ×28 个像素点,共含有 6 万张训练图片和 1 万张测试图片。在本实验中,Lenet 神经网络模型调用该数据库的数据。

ImageNet 是一个超过 1500 万张被标记的高分辨率的图像数据库,属于约 2.2 万个不同类别。这里采用的是 2012 年 ImageNet 大规模视觉识别挑战(ILSVRC2012)提供的数据集作为实验数据集,这是 ImageNet 的一个子集,包含 120 万个训练样本和 5 万个验证样本,归属于 1000 个类别之中。

6.3 硬件特征

利用 Verilog 语言在 RTL 级实现了本实验的加速器,并且利用 Synopsys Design Compiler 工具以标准 VT 中的 TSMC 65nm GP 工艺进行综合,并利用 Synopsys ICC 编译器进行布局和布线。然后,使用 Synopsys VCS 工具进行了模拟和验证设计,并基于模拟的 Value Change Dump(VCD)文件使用 Synopsys Prime-Time PX 来估算功耗。

为了公平起见,加速器采用 Cambricon-X 相同的配置,包括频率、片上面积和峰值性能,即 1 GHz, 56 kB 的片上 SRAM 和 16 个运算单元,本文设计的加速器的峰值可达到的每个时钟周期完成 768 个运算操作。加速器的布局特征如表 2 所示,在本加速器的面积总共为 8.04 mm^2,为 Cambricon-X 的 1.26 倍;功率为 2 370.95 mW,是 Cambricon-X 的 2.49 倍。

表 2 加速器的布局特性

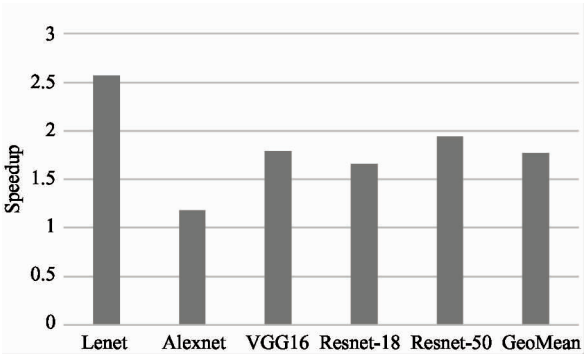
加速器	面积(mm^2)	占比(%)	功率(mW)	占比(%)
整体	8.04	100	2370.95	100
CP	0.20	2.49	86.32	3.64
SBin	0.55	6.84	93.32	3.94
SBout	0.55	6.84	93.32	3.94
FUs	6.74	83.84	2 097.99	88.49
FU				
--CU	0.08	0.99	47.79	2.02
--PB	0.07	0.87	9.49	0.40
--DC	0.27	3.36	73.84	3.11

6.4 性 能

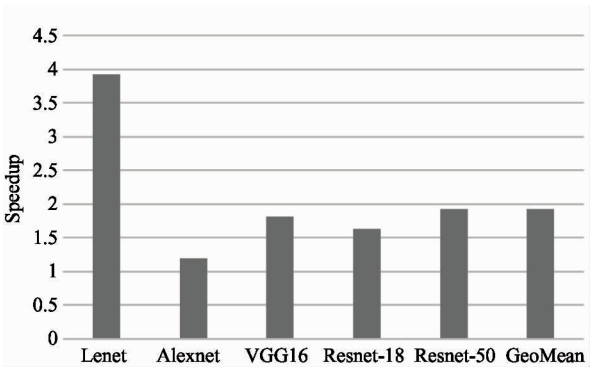
本文将加速器性能和 Cambricon-X 进行了对比。如图 8(a)所示,各网络的整体性能达到了1.77 倍的加速比。其中加速比最高的为 Lenet 网络模型,约为 2.57 倍,因为该网络结构中计算量最大的为第一层卷积运算,加速器的设计能够充分利用 Mnist 数据集的近 20% 的稀疏度,从而大幅缩短 Lenet 的运算时间。

进一步,对卷积层进行了对比和分析,如图 8(b)所示,其表现基本和整体网络的加速比趋势相同,这是因为卷积层的运算是神经网络中主要运算,占用了大部分的运算时间。而卷积层又是计算密集型运算层,利用神经元稀疏提高运算的有效性,能够大大加速卷积层的运算,其平均的加速比为1.93。

本实验还对各 block 的 BN 层进行了对比,如图 9所示,Resnet-18 和 Resnet-50 的平均分别达到了 4.40和 4.57 倍加速比。这是由于加速器能够有效完成逐元素加和逐元素乘等操作,从而进行 BN 层的运算时具有更高的加速比。



(a) 网络模型整体加速比



(b) 卷积层加速比

图 8　本加速器和 Cambricon-X 的性能比较

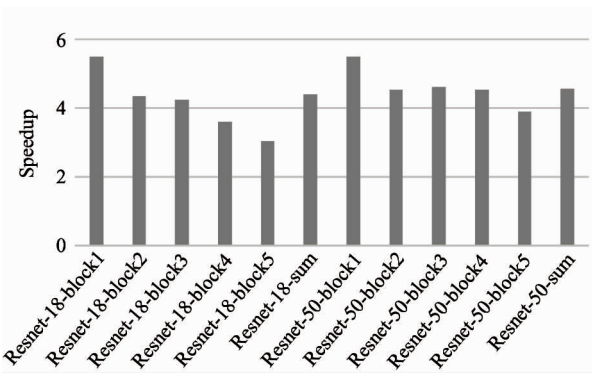


图 9　Resnet-18 和 Resnet-50 中各 block 中的 BatchNorm 层的性能比较

6.5　功　耗

本文对加速器的能耗开销和 Cambricon-X 进行了对比,如图 10 所示。此处,采用和 Cambricon-X 相同的调度方式,功耗开销和 Cambricon-X 的功耗开销基本相同。这是因为功耗开销最大的是卷积层和全连接层的访存开销,在相同的调度方式下,其加速器和 Cambricon-X 的开销基本相同。另外,虽然加速器的功率比 Cambricon-X 高,但是运算时间较短,其整体功耗开销也基本相同。其中,由于大大缩短了运算时间,所以 Lenet 功耗节省最多,仅为 Cambricon-X 的 95%。

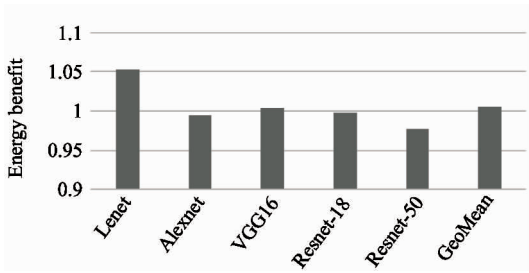


图 10　本加速器和 Cambricon-X 的功耗比较

7　结　论

本文提出的神经网络加速器架构,能够同时支持神经元稀疏和权值稀疏,从而能够进一步增加运算效率;增加逐元素运算的功能,既能够有效加速 Lenet 等神经网络模型中的各层的相关运算,也可以支持 Resnet 等新型神经网络模型中的各层的相关运算。实验结果标明,基于 5 个神经网络基准在推测阶段的运算性能的评估,本文提出的加速器和先进的稀疏神经网络加速器相比有 2.57 倍的加速比,其中对 Resnet-18 和 Resnet-50 的 BN 层平均分别有 4.40 倍和 4.57 倍的加速比。

参考文献

[1] Fu J, Zheng H, Mei T. Look closer to see better: recurrent attention convolutional neural network for fine-grained image recognition[C]. In: Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 2017. 4476-4484

[2] Venugopalan S, Rohrbach M, Donahue J, et al. Sequence to sequence——video to text[C]. In: Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 2015. 4534-4542

[3] Abdel-Hamid O, Mohamed A R, Jiang H, et al. Convolutional neural networks for speech recognition[J]. *IEEE/ACM Transactions on Audio Speech & Language Processing*, 2014, 22(10):1533-1545

[4] Eriguchi A, Hashimoto K, Tsuruoka Y. Tree-to-sequence attentional neural machine translation[J]. *arXiv*: 1603.06075, 2016

[5] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]. In: Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, USA, 2012: 1097-1105

[6] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. *arXiv*: 1409.1556v6 , 2014

[7] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016. 770-778

[8] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting [J]. *Journal of Machine Learning Research*, 2014, 15(1):1929-1958

[9] 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

[10] Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines[C]. In: Proceedings of the International Conference on International Conference on Machine Learning, Haifa, Israel, 2010. 807-814

[11] Lécun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. *Proceedings of the IEEE*, 1998, 86(11):2278-2324

[12] Chen Y, Chen Y, Chen Y, et al. DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning[C]. In: Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, Salt Lake City, USA, 2014. 269-284

[13] Chen Y, Luo T, Liu S, et al. DaDianNao: a machine-learning supercomputer [C]. In: Proceedings of the IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014. 609-622

[14] Du Z, Fasthuber R, Chen T, et al. ShiDianNao: shifting vision processing closer to the sensor[C]. In: Proceedings of the International Symposium on Computer Architecture, Portland, USA, 2015. 92-104

[15] Chi P, Li S, Xu C, et al. PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory[C]. In: Proceedings of the ACM/IEEE International Symposium on Computer Architecture, Seoul, Korea, 2016. 27-39

[16] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: a convolutional neural network accelerator with in-Situ Aanalog arithmetic in crossbars[J]. *ACM Sigarch Computer Architecture News*, 2016, 44(3):14-26

[17] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural networks [C]. In: Proceedings of the Advances in Neural Information Processing Systems, Montreal, Canada, 2015. 1135-1143

[18] 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

[19] Albericio J, Judd P, Hetherington T, et al. Cnvlutin: ineffectual-neuron-free deep neural network computing[C]. In: Proceedings of the International Symposium on Computer Architecture, Seoul, Korea, 2016. 1-13

[20] Zhang S, Du Z, Zhang L, et al. Cambricon-X: an accelerator for sparse neural networks[C]. In: Proceedings of the IEEE/ACM International Symposium on Microarchitecture, Taipei, China, 2016. 1-12

[21] Parashar A, Rhu M, Mukkara A, et al. SCNN: an accelerator for compressed-sparse convolutional neural networks[C]. In: Proceedings of the ACM/IEEE 44th Annual International Symposium on Computer Architecture, Toronto, Canada, 2017. 27-40

[22] Jia Y Q, Shelhamer E, Donahue J, et al. Caffe: convolutional architecture for fast feature embedding[J]. *arXiv*:1408.5093, 2014

[23] Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift [C]. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, 2015. 448-456

[24] Du Z, Lingamneni A, Chen Y, et al. Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators[C]. In: Proceedings of the 2014 19th Asia and South Pacific Design Automation Conference, Singapore, 2014. 201-206

An accelerator for sparse neural network

Zhou Shengyuan[*,**,***], Du Zidong[*], Chen Yunji[*,**]

([*] State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] Cambricon Tech. Ltd, Shanghai 201306)

Abstract

A novel sparse neural network accelerator architecture is proposed for the growing neural network scales and the ever-changing structures of neural network models. The architecture can effectively leverage the synapse sparsity and the neuron sparsity to further improve the performance of the accelerators when processing sparse neural networks. At the same time, it supports element-wise multiplication/addition so that it further enhances the flexibility, supports and accelerates new neural network structures efficiently such as Resnet. The experiment results show that, based on the five representative neural network models, the proposed accelerator outperforms the state-of-the-art sparse neural network accelerator 2.57x on performance. Especially, it achieves 4.40x and 4.57x speedup on average on the BN layers of Resnet-18 and Resnet-50, respectively.

Key words: neural network, sparse neural network, accelerator