高技术通讯 2022 年 第32 卷 第7 期:696-707

doi:10.3772/j.issn.1002-0470.2022.07.004

# 深度卷积的软硬件协同优化设计与实现①

齐 豪②* 刘少礼** 李 威③***

(*中国科学技术大学计算机科学与技术学院 合肥 230026)

(**上海寒武纪信息科技有限公司 上海 201306)

(***中国科学院计算技术研究所处理器芯片国家重点实验室 北京 100190)

**摘 要** 近年来,深度学习技术被广泛应用。由于移动设备同时受到算力和功耗的限制,很多轻量级的网络被提出,比如 Xception、MobileNet 系列等。在这些轻量级网络中,深度卷积的层数占网络中所有卷积层数的 31% ~50%,故如何优化深度卷积的运算是一个值得研究的问题。通用中央处理器(CPU)、固定运算器长度的单指令多数据流(SIMD)处理器均无法高效处理神经网络中的各种规模的深度卷积,性能较低。针对这一问题,本文提出了一种软硬件结合的方法优化深度卷积的计算,通过一个多种权值传输模式的硬件架构设计,结合软件模式选择、数据拆分等优化方式,在提高运算效率的同时减少了访存量。实验结果表明,使用该方法实现的深度卷积加速器,相比通用 CPU 最大可达 9.3 倍的性能加速,相比运算器长度为 64 的单核 SIMD 处理器最大可达 29.3 倍的性能加速。

**关键词** 神经网络;深度卷积;加速器;软硬件协同优化;计算效率

## 0 引 言

近年来,深度学习技术在图像分类[1]、语音识别[2]、自然语言处理[3]、视频描述生成[4]等多个领域得到广泛的应用并取得巨大的成功,很多经典的神经网络模型被提出,比如 AlexNet[5]、VGGNet[6]、ResNet[7]和 Inception V3[8]等。这些模型在提升算法精度的同时,其深度和规模都在不断增加,其计算量和访存量也在迅速增加。研究人员从算法优化和硬件加速维度提出了多种优化方法来解决上述问题,其中算法优化包括模型剪枝[9]、参数量化[10]等,硬件加速结构包括 DianNao[11]、DaDianNao[12]等。

移动设备同时受到算力和功耗的限制,因此针对移动设备的研究致力于设计更轻量级的模型,这些模型由较少的层和计算操作组成,比如 Xception[13]、MobileNetV1[14]、MobileNetV2[15]、MobileNetV3[16]和 EfficientNet[17]等。在这些轻量级模型中,深度卷积(depthwise convolution)是一个比较常用的计算操作,比如在 Xception 网络的所有卷积操作中,深度卷积操作的数量占 50%。因此,如何加速深度卷积的计算是一个值得研究的问题。

深度卷积和传统卷积的不同之处在于它的输入通道个数和输出通道是相同的,并且是一一对应的,输入通道之间不累加。目前处理深度卷积的通常做法是将数据维度设置为 *NHWC*,其中 *N* 表示特征图的个数,*H* 和 *W* 分别表示特征图的长度和宽度,*C* 是通道数,然后使用单指令多数据流(single instruction multiple data,SIMD)指令进行处理[18]。这种处理方式存在一个问题,即如何设计相应处理器的运算器长度?如果运算器的长度太长,对于 *C* 较小的

① 国家重点研发计划(2020AAA0103802),国家自然科学基金(61732002,61925208,61732007,61906179,U20A20227),中国科学院战略性先导科技专项(XDBS01050200),北京智源人工智能研究院和中国科学院青年创新促进会和科学探索奖资助项目。

② 男,1996 年生,硕士生;研究方向:计算机体系结构;E-mail:theqihao@mail.ustc.edu.cn。

③ 通信作者,E-mail:liwei2017@ict.ac.cn。

(收稿日期:2021-03-10)

深度卷积,运算效率较低,比如 MobileNetV3 的深度卷积最小 $C$ 是 16,如果用运算器长度为 64 的 SIMD 处理器计算,运算效率只有 25%;如果运算器长度设置为 16,虽然计算效率可以达到 100%,但此时处理器的计算峰值偏低。故一个固定运算器长度的 SIMD 处理器不能够高效地处理各种规模的深度卷积。

针对这一问题,本文提出了一种软硬件结合的方式来优化深度卷积的计算。本文关注于解决各种规模的深度卷积的优化,通过设计特定的数据放置方式和多种数据传输模式,结合软件模式选择、数据拆分等优化方式,在提高运算效率的同时减少了访存量,提升了算法实现性能。本文主要贡献如下。

(1) 提出一种新型的深度卷积加速器(depth-wise convolution accelerator,DCA)结构,该加速器设计多种权值传播模式以高效支持各种规模的深度卷积。

(2) 基于本文设计的深度卷积加速器,在软件层面,选择使用不同的权值传输模式和数据拆分方式,充分发挥出深度卷积加速器的性能。

(3) 实施了 DCA 加速器和软件优化,并测试其在不同深度卷积规模上的实现性能。实验结果表明,相比运算器长度为 64 的单核 SIMD 处理器最大可达 29.3 倍的性能加速。

## 1　相关工作

近年来,神经网络飞速发展,计算量和访存量的迅速增加给计算机系统带来了巨大的挑战。中央处理器(central processing unit, CPU)和图形处理器(graphics processing unit, GPU)为了保证其通用性,无法以较高的性能功耗比处理神经网络任务。因此,神经网络加速器应运而生并得到了迅速的发展。Chen 等人[11]在 2014 年提出了 DianNao,DianNao 着重分析了单核深度学习处理器片上存储层次以及访存子系统的设计对性能和功耗的影响,专注于片上存储层次和访存子系统的优化。DaDianNao[12]是一种用于高效处理大规模深度学习算法的多核架构,并且采用片上 eDRAM 提高片上存储密度,完全避免了片外访存的开销。ShiDianNao[19]是为以极低功耗开销快速处理嵌入式系统中的图像应用而设计的架构,它通过将整个卷积神经网络(convolutional neural networks,CNN)模型保存到片上存储器内,并放置到图像传感器旁边,直接接收图像传感器的输入,可以完全消除系统对片外存储器的访问。文献[20]设计了一款可以支持多种机器学习算法的运算单元,该运算单元实现了低面积、低功耗的需求。文献[21]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。文献[22]提出了一种软硬件协同的方法,有效解决了稀疏神经网络的稀疏不规则性问题。MW-DLA[23]是一种支持动态可配置数据宽度的深度学习加速器,可支持多种数据格式。为了提高神经网络加速器的灵活性和通用性、提升代码密度,Liu 等人[24]提出了神经网络指令集 Cambricon。

随着神经网络加速器的不断发展,许多软件层面的支持和优化被提出来。DLPlib[25]是一个基于深度学习处理器的高性能库,该库支持主流神经网络算法的推理操作和一组基本的矩阵和向量操作。ZhuQue[26]定义了神经网络数据的动态标签和静态标签,提出了一种神经网络数据分类方法 NNData-Class,实现了基于标记数据布局的神经网络开发套件 LDL-NDK,最后抽象出了基于标记的神经网络编程模型。文献[27]提出了一系列针对神经网络加速器的编译优化技术,包括图优化、循环展开和流水线优化等。TVM [28]是一种深度学习编译器框架,可以解析各种前端框架的计算图。它使用 Halide[29]中间表示(intermediate representation,IR)表示计算循环并提供多种级别的优化。文献[18]提出了一种基于 ARM 处理器的深度卷积优化方案。

上述工作都是针对传统卷积、矩阵乘和其他机器学习算法等进行硬件支持或软件优化的工作,没有针对深度卷积进行深入分析和探讨。目前多数平台通过使用固定运算长度的 SIMD 处理器来实现深度卷积,但是这种方法具有局限性,即固定运算长度的 SIMD 处理器对计算 $C$ 方向很小的深度卷积的效

率不高,从而可能会影响最终性能。本文提出的深度卷积加速器能够有效计算各种规模的深度卷积算法,并且可根据硬件的设计进行进一步的软件优化。

## 2 深度卷积

本节主要介绍深度卷积的运算和访存特征,以及在实际神经网络当中的规模特征。

### 2.1 深度卷积计算和访存特征

深度卷积中输入特征图的通道数和输出特征图的通道数是相同的,并且各个输入特征图的通道之间不会像传统卷积算法那样进行累加操作,各个通道独立运算。如图 1 所示,输入 ***input*** 、权重 ***weight*** 及输出 ***output*** 均为三维向量,其 3 个维度分别对应高度($h$)、宽度($w$)及通道($c$)。其中,输入 ***input*** 在 3 个维度上的大小是 $hi \times wi \times ci$;输出 ***output*** 在 3 个维度上的大小是 $ho \times wo \times co$, 其中 $co$ 和 $ci$ 是相同的;深度卷积的权值 ***weight*** 在 3 个维度上的大小是 $ky \times kx \times ci$, 其中 $ky$、$kx$ 分别表示卷积核在 $H$、$W$ 方向的大小。假设深度卷积在 $H$ 和 $W$ 方向的步长 *stride* 分别为 $sy$ 和 $sx$,则深度卷积运算可以形式化地表示为式(1)。

$$\boldsymbol{output}(h, w, c) = \sum_{i=0}^{ky-1}\sum_{i=0}^{kx-1} \boldsymbol{input}(h \times sy + i, w \times sx + j, c) \times \boldsymbol{weight}(i, j, c) \tag{1}$$

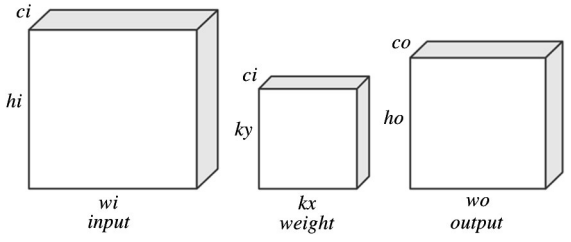


**图 1 深度卷积**

陈云霁等人[[30]]在《智能计算系统》一书中提到神经网络访存有两个重要的特性,分别是可解耦性和可重用性。在加速器的设计中需要考虑这两个特性。

深度卷积的乘加运算器(multiply-accumulator, MAC)计算量和访存量可由式(2)和式(3)得到。对于 MobileNetV1、MobileNetV2、MobileNetV3 网络中的所有深度卷积规模,计算量和访存量的比例平均为 3.8∶1。加速器的计算和访存是可以并行的,因此在设计深度卷积加速器时要着重考虑算力和访存带宽之间的均衡,应尽量让计算时间和访存时间相等。

$$\text{MAC 计算量} = ho \times wo \times co \times ky \times kx \tag{2}$$

$$\text{MAC 访存量} = (hi \times wi + ho \times wo + ky \times kx) \times co \tag{3}$$

### 2.2 深度卷积规模特征

MobileNet V1、MobileNet V2、MobileNet V3 中的深度卷积的输入神经元规模有一个特征,即在网络的前几层,$C$ 方向值较小,而 $HW$ 方向值较大;在网络的后几层,$C$ 方向值较大,而 $HW$ 方向值较小。

网络中输入神经元的 $C$ 方向的大小取值范围是[16, 1024]。$C$ 方向最小是 16,如果使用一个运算器长度为 64 的 SIMD 处理器来计算,实际计算效率最多只有 25%。如果使用一个运算器长度为 16 的 SIMD 处理器来计算,计算效率虽然可以达到 100%,但此时处理器算力相对较小。所以在设计加速器时,要选取合适的运算器规模。

## 3 深度卷积加速器

基于上一节对深度卷积的分析,本节提出了一个新型的深度卷积加速器架构,称之为 DCA(depth-wise convolution accelerator)。该架构通过设计特定的数据放置方式和多种数据传输模式,高效完成各种规模的深度卷积运算。

### 3.1 加速器整体架构

加速器的整体架构如图 2 所示,主要包括以下几个部分:1 个矩阵运算单元(matrix functional unit,

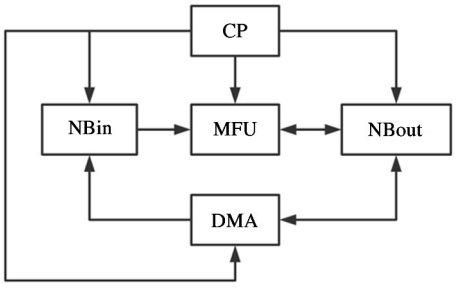


**图 2 深度卷积加速器总体架构**

MFU)、1 个控制处理器(control processor,CP)、2 个片上缓存(input neuron buffer,NBin 和 output neuron buffer,NBout)和直接存储访问模块(direct memory access,DMA)。

### 3.2　存储单元

加速器的 2 个片上缓存分别命名为 NBin 和 NBout,其中 NBin 存放输入数据,NBout 存放权值和输出数据。加速器没有为权值数据单独分配一个缓存,而是和输出数据共享一个缓存,主要有以下 2 个原因:

(1) 权值数据规模相对于输入和输出数据规模较小,可以和输入或输出数据共享一个缓存,以减少单独为权值分配缓存的硬件开销。

(2) 权值数据将以不同的方式传输到各个运算单元中(详见 3.4 节),并且与输入数据进行计算,故输入数据和权值数据不能放在同一个缓存上。

### 3.3　运算单元

矩阵运算单元由多个处理单元(processing element, PE)组成,PE 之间使用胖树(FatTree)进行连接,以避免线路拥塞,如图 3 所示。不同于 NBout,NBin 被分成了多份,每个 PE 包含了部分 NBin。每个 PE 内部结构如图 4 所示,包含以下几个部分:1 个运算器(computational unit,CU)、1 个数据控制器(data controller,DC)和 2 个私有的片上缓存(private buffer,PB),其中 1 个缓存用于存放神经元(input private buffer ,INPB),另外 1 个缓存用于存放权值(weight private buffer ,WPB)。

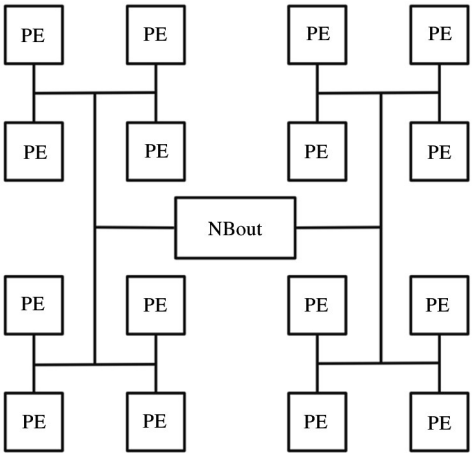


图 3　MFU 示意图

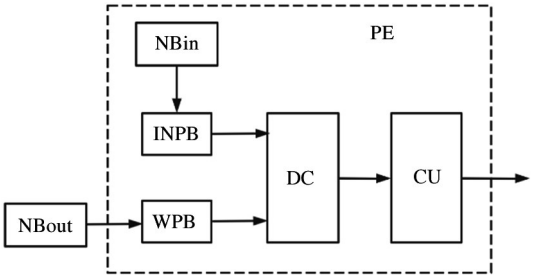


图 4　PE 内部结构图

CU 是一个向量 MAC 单元,每个向量 MAC 包含多个标量 MAC 运算器。MAC 接收 16 位定点数进行计算而不是常用的 32 位单精度浮点数,因为研究表明,使用 16 位定点数即可保证深度学习算法的精度基本不受影响[11]。

### 3.4　数据通路

数据通路的设计主要有以下 3 个特点:(1) 驻留输入、传输权值;(2) 多种权值传输模式;(3) 多播部分输入数据。

驻留输入、传输权值。对于实际的深度卷积,其权值相对于输入较小。故本文在设计加速器时,与 DaDianNao[12] 的驻留权值、传输输入的方式不同,本文的加速器将输入驻留在与运算器相邻的片上缓存上,传输权值。这样做的目的是减少数据传输带来的性能和功耗开销。

多种权值传输模式。为了支持不同的深度卷积规模,加速器设计了 3 种权值传播模式,分别是权值广播、权值多播、权值单播,如图 5 中(a)、(b)、(c)所示,不同的线型分别代表不同的权值。

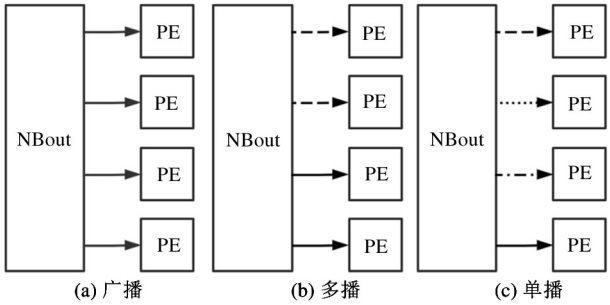


图 5　权值传输模式示意图

加速器在处理一次任务时,需要将当前任务拆分到每一个 PE 上。任务在 PE 之间的拆分可以看作深度卷积的输入数据在 $H$、$W$、$C$ 3 个方向的拆分。

根据任务在 PE 之间不同的拆分方式，PE 之间存在多种共享权值的方式，对应了不同的权值传输模式。在神经网络前几层，*C* 方向值较小、*HW* 方向值较大，此时 PE 在 *HW* 方向拆分输入，每个 PE 使用相同的权值，对应的权值传输模式是权值广播，如图 5(a)所示；在神经网络后几层，*C* 方向值较大、*HW* 方向值较小，此时 PE 在 *C* 方向拆分输入，对应的权值传输模式是权值单播，每个 PE 使用不同的权值，如图 5(c)所示；在网络的中间几层，*C* 方向大小适中，此时 PE 在 *HWC* 3 个方向拆分输入，对应的权值传输模式是权值多播，如图 5(b)所示。权值多播又分为多种形式，将 PE 分组，每一组内使用相同权值，组之间使用不同的权值，可以有 2 个 PE 为一组、4 个 PE 为一组、8 个 PE 为一组等多种模式。

多播部分输入数据。在权值广播和权值多播模式中，输入数据会在 *HW* 方向拆分到不同 PE 的 NBin 上。根据卷积的计算特征，对于卷积核大于 *stride* 的情况，*HW* 方向相邻的输出数据点会使用相同的输入数据，所以会出现需要 2 个 PE 内部的 NBin 存放部分相同输入数据的情况。图 6 给出了相同输入数据在 *HW* 平面拆分到不同 PE 的情况，图中 *input* 数据中灰色的部分是 2 个 2×2 的输出大小对应的重叠的输入数据，其中卷积核和 *stride* 的大小分别是 3 和 1。在这种情况下，本文的加速器专门设计了一种部分数据多播的通路，以加速加载输入数据有冗余的情况。数据通路会检查从 DMA 传输到 NBin 的数据，如果是冗余数据，那么进行多播。

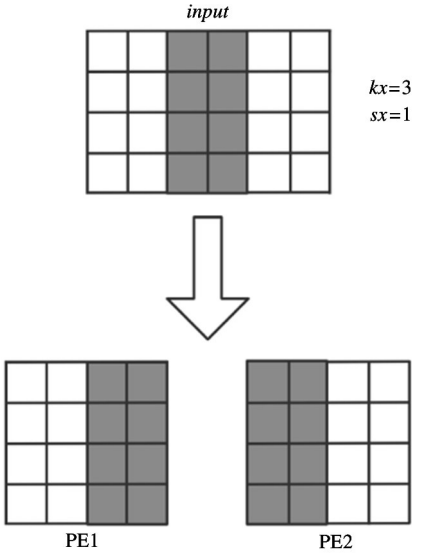


**图 6 PE 之间共享数据示意图**

## 4 软件实现和优化

本节介绍针对本文设计的 DCA 加速器，软件应该如何实现并且如何优化。本节的软件实现是在 DLPlib[25] 的基础上进行扩展，使其支持 DCA 加速器。其中扩展包括权值传输模式的选择、数据拆分。

### 4.1 模式选择

本文的加速器设计了多种权值传输模式，在使用时需根据具体的情况选择不同的权值传输模式。在权值传输模式选择时，考虑的因素有以下 3 个方面。

(1) 运算效率。在计算时，运算效率由有效工作的 PE 占所有 PE 的比例和每个 PE 中有效工作的运算器比例所决定。假设每个 PE 做有效工作的运算器的比例都是相同的，则运算效率 = 有效工作的 PE 比例 × 每个 PE 中有效工作的运算器的比例。

(2) 访存性能。考虑访存局部性，尽量减少跳跃访存(非连续内存访问)，因为跳跃访存可能会导致存储器内部行缓冲区命中率低，进而降低访存性能。

(3) 输入数据冗余。输入数据冗余分为 2 种类型，即输入数据在片外存储器上拆分的冗余和输入数据在 PE 之间拆分的冗余。第 1 种冗余是因为片上缓存大小限制，输入数据在片外存储器拆分，会引入额外的访存或者片上数据传输开销，从而可能影响整体性能。第 2 种冗余不仅会占用更多片上缓存空间，而且冗余的数据写入缓存会增加 DCA 加速器的功耗。其中第 2 种冗余数据的访存行为可以通过本文 3.4 节介绍的多播部分输入数据的方式消除。但是第 2 种冗余数据的增多会导致其占用片上缓存增多，使得输入数据在片外存储器的拆分粒度变小，拆分次数增加，最终导致第 1 种数据冗余规模增加，整体性能下降。

为保证访存的性能，本文在做数据拆分时，不会优先拆分 *C* 方向，即尽量减少跳跃访存。因此，后面的内容主要考虑运算效率和数据冗余对模式选择

的影响。

本文的重点在于加速深度卷积的性能,即最小化深度卷积任务在加速器上的运行时间。因为DCA加速器的计算和访存可以并行,所以在理想情况下,最终运行时间 $T$ 可根据计算时间 $T_{cpt}$ 和访存时间 $T_{io}$ 确定,即:

$$T = \max(T_{cpt}, T_{io}) \tag{4}$$

假设任务的计算量、访存量和输入数据冗余量分别为 $CPT$(MAC)、$IO$(B)和 $R$(B),加速器的计算峰值和带宽分别为 $CP$(Giga operations per second, GOPS)和 $BW$(GB/s),任务运行在加速器上的计算效率为 $CE$(computing efficiency),则有:

$$T_{cpt} = \frac{CPT}{CE \times CP} \tag{5}$$

$$T_{io} = \frac{IO + R}{BW} \tag{6}$$

从式(5)和式(6)可以看出,在任务规模以及加速器性能参数确定的情况下,软件层面可以控制并且影响任务最终性能的2个因素分别是计算效率 $CE$ 和输入数据冗余量 $R$。

假设PE个数 $M$ 为16,向量MAC的长度为16。因为每个PE的向量MAC是沿 $C$ 方向进行处理,所以计算效率主要受 $C$ 方向大小的影响。图7给出了不同权值传输模式下,计算效率随着 $C$ 方向大小变化而变化的示意图,其中在权值多播模式下,4个PE为一组(为简便起见,后续权值多播模式都使用4个PE一组)。可以看出,权值广播模式的计算效率在多数情况是很高的,并且随着$C$的不断增大趋于稳定;权值单播模式在多数情况下计算效率不高,并且很不稳定;权值多播模式的计算效率适中并且变化幅度较小,随着 $C$ 的增大逐渐增大并趋于稳定。因此,若只从计算效率的角度考虑,应该优先选择权值广播模式,其次是权值多播模式,最后是权值单播模式。但是权值广播和多播模式在PE之间会有部分输入数据的冗余,会影响最终性能。因此,在选择权值传输模式时还需要进一步考虑输入数据冗余对性能的影响。

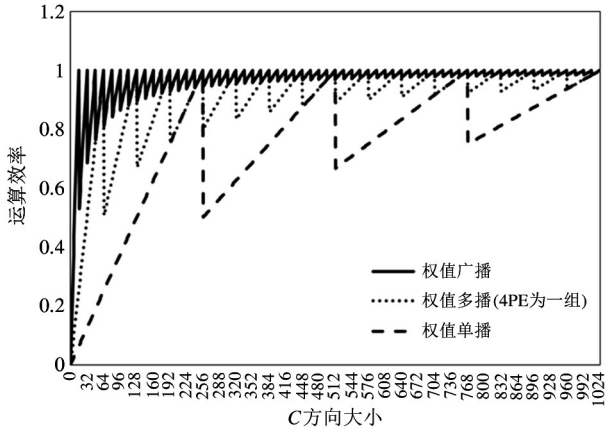


**图7　不同模式下的计算效率**

图8给出了在不同传输模式下,数据冗余量随着 $C$ 方向的变化曲线,其中 $HW$ 方向的输出大小都是1024,卷积核和 *stride* 分别是3和1。从图8可以看出,随着 $C$ 方向逐渐增大,3种模式的数据冗余量都逐渐增多,其中权值广播的变化幅度较大,权值单播模式的变化幅度较小,权值多播的变化幅度介于二者之间。权值广播模式的曲线在 $C$ 为896之后消失,因为在 $C$ 方向不拆分的情况下,此时每个PE的片上缓存大小小于输入数据在 $HW$ 方向拆分最小粒度所需的缓存空间大小。

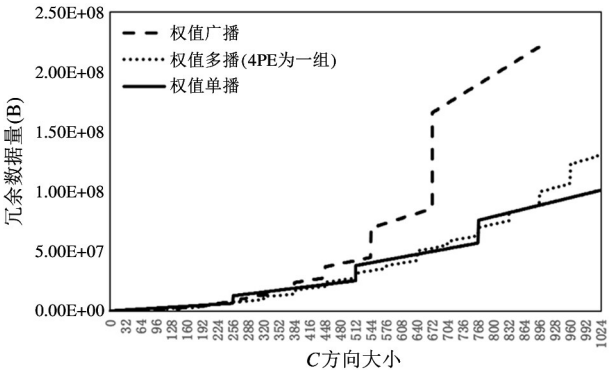


**图8　不同模式下的数据冗余量**

假设运算单元的时钟频率为500 MHz,NBin的大小为0.25 MB,内存访问带宽为64 GB/s,此时加速器是计算量决定性能。图9给出了在不同权值传输模式下,任务运行时间 $T$ 随着 $C$ 方向增大而变化的曲线。从图9可以看出,在多数情况下,因为权值广播的计算效率较高,权值广播模式的运行时间最小。此外还可以看出3种模式的运行时间随着 $C$ 方向的增大呈现阶梯形状,并且权值单播模式下变化幅度最大,其次是权值多播和广播模式。如果仅从任务运行时间来选择模式,应该优先考虑权值广播模

式,因为在多数情况下,权值广播运行时间最小。

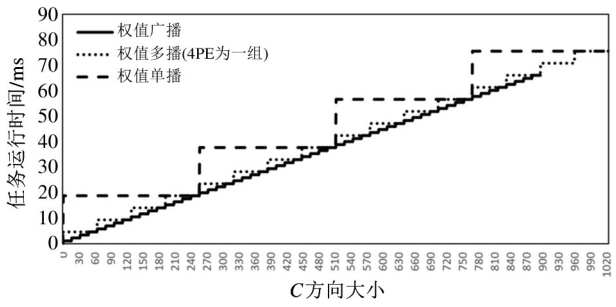


**图 9　任务在计算量决定性能时,不同模式下的运行时间**

图 10 给出了任务在计算量决定性能时,不同模式下的最小运行时间,在运行时间相同的情况下,优先考虑数据冗余量小的模式。从图 9 和图 10 可以看出,在权值广播和多播的数据规模和计算效率相同的情况下,应选择使用权值多播模式,比如 $C$ 大于 48 且小于等于 64 的情况,因为此时权值广播的数据冗余量比权值多播模式大。

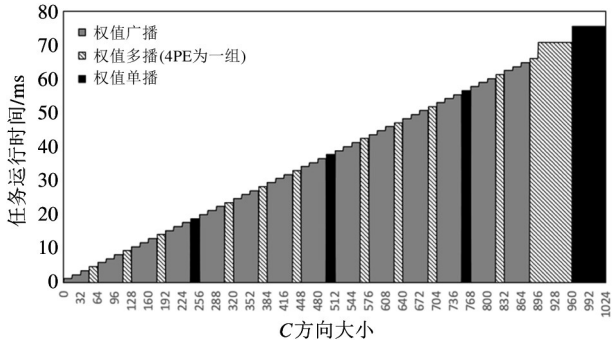


**图 10　任务在计算量决定性能时,不同模式下的最小运行时间**

PE 之间在 $C$ 方向拆分不会引入冗余数据,故冗余数据的大小只需要考虑 $HW$ 2 个方向。假设一次计算的输出数据在 $HW$ 方向的大小分别是 $ho$、$wo$,输出的 $HW$ 平面在权值广播和权值多播模式下分别拆分成 $x$ 和 $y$ 份。因为权值多播在 $C$ 方向拆分,所以在 $HW$ 方向拆分的个数 $y$ 小于 $x$。为了 PE 之间的计算负载均衡,每个 PE 应该处理相同规模的数据,则权值广播和权值多播在 PE 拆分之后的输出在 $HW$ 方向大小的乘积分别为 $ho \times wo/x$,$ho \times wo/y$。根据 4.2 节数据拆分的结论,即每个 PE 处理输入的 $HW$ 方向大小应尽可能相等。因此,权值广播和权值多播模式下的每个 PE 在 $HW$ 方向的大小分别为 $\sqrt{ho \times wo/x}$ 和 $\sqrt{ho \times wo/y}$。

在卷积核和 *stride* 分别为 3 和 1 的情况下,权值广播的 $HW$ 平面的数据冗余量如式(7)最左端所示,权值多播的数据冗余量类似。因为输出大小和卷积参数都是已知量,权值广播模式的数据冗余量可以表示为 $C_1 \times x + C_2 \times \sqrt{x} + C_3$,同理可得权值多播的数据冗余量为 $C_1 \times y + C_2 \times \sqrt{y} + C_3$。因为 $x > y$,显然 $C_1 \times x + C_2 \times \sqrt{x} + C_3 > C_1 \times y + C_2 \times \sqrt{y} + C_3$。所以在数据规模确定的情况下,权值广播的数据冗余量比权值多播模式大。因此,在任务规模和计算效率相同的情况下,优先考虑使用权值多播模式而不是权值广播模式。

$$x \times \sqrt{(ho \times wo/x)} + ky - 1) \times (\sqrt{(ho \times wo/x)} + kx - 1) - (ho + ky - 1) \times (wo + kx - 1) = x \times (ky - 1) \times (kx - 1) + \sqrt{ho \times wo \times x} \times (ky + kx - 2) + ho \times wo - (ho + ky - 1) \times (wo + kx - 1) = C_1 \times x + C_2 \times \sqrt{x} + C_3 > C_1 \times y + C_2 \times \sqrt{y} + C_3 \tag{7}$$

在权值单播模式下,PE 之间在 $C$ 方向拆分,PE 之间没有冗余数据。因此,当权值单播和权值多播、权值广播的计算效率相同时,应选择权值单播模式,比如图 10 中 $C$ 大于 240 且小于等于 256 的情况。

结合式(4)~(6)以及对其的分析,本文以最小化任务在加速器上的运行时间为目标,然后评估任务在不同模式下的运行时间,最终选择运行时间最小的模式。如果运行时间相同,应尽可能减小 $T_{cpt}$ 和 $T_{io}$。如果 $T_{cpt}$ 和 $T_{io}$ 也相同,那么应该减少片上数据传输量,即减少数据冗余量。

### 4.2　数据拆分

数据拆分主要分为 2 类:因片上缓存大小限制而导致的拆分和输入数据在 PE 之间的拆分。第 1 类是因为片上缓存 NBin 和 NBout 的大小是有限的,当深度卷积的规模较大,即数据大小大于缓存大小,需要对数据进行拆分。其中包括权值拆分、输入和输出拆分。权值拆分的情况比较少,这里不做讨论。第 2 类是输入数据在 PE 之间的拆分,因为在权值广播和多播模式下,PE 之间的输入数据会有冗余。软件在做拆分时,应尽可能地减少冗余数据。在确

定一次计算的大小时,需要同时考虑以上 2 类拆分。具体步骤如下。

输入和输出在缓存上的拆分。对于输入和输出的拆分,首先考虑输出的拆分,然后根据输出大小以及卷积参数推测出输入的大小。考虑深度卷积的数据访存局部性,拆分的次序分别是 $ho$、$wo$、$co$,应尽量保证 $C$ 方向不拆分。

输入数据在 PE 之间的拆分。针对不同的权值传输模式,输入数据在 PE 之间的拆分将分为 3 类,即 $HW$ 方向拆分、$HWC$ 方向拆分和 $C$ 方向拆分。相应的硬件模式是权值广播、权值多播、权值单播。

输入数据在 $HW$ 方向拆分时,PE 之间会有部分冗余输入数据,如图 6 所示。相同大小的输入数据在 PE 之间的拆分方式不同,冗余数据大小不同,输入数据占用片上缓存空间大小也不同。因此,在输入数据规模确定的情况下,如何在 PE 之间拆分输入数据以最小化片上缓存空间是个值得研究的问题。

为了 PE 之间的计算负载均衡,每个 PE 的计算量应尽可能相同。假设输出数据在 $HW$ 方向的大小分别是 $ho$ 和 $wo$,$stride$ 在 $HW$ 方向都为 1,拆分后每个 PE 的输出数据在 $HW$ 方向的大小是 $hp$ 和 $wp$,PE 数量是 $M$,则 $hp \times wp$ 和 $ho \times wo/M$ 相等。每个 PE 的输入数据大小可以根据输出大小和深度卷积参数推测出,如式(8)所示。

$$\begin{aligned}(hp + ky - 1) \times (wp + kx - 1) &= (hp \times wp) + hp \times (kx - 1) + wp \times (ky - 1) + (ky - 1) \times (kx - 1) = \\ C_4 \times (hp + wp) + C_5 &\geqslant 2 \times C_4 \times \sqrt{hp \times wp} + C_5 \\ &= 2 \times C_4 \times \sqrt{ho \times wo/M} + C_5 \end{aligned} \tag{8}$$

在式(8)中,$C_4 = hp \times wp + (ky - 1) \times (kx - 1)$,因为 $hp \times wp$ 和 $ho \times wo/M$ 相等,在输出数据大小确定的情况下,$hp \times wp$ 是已知量;因为 $ky$、$kx$ 都是已知量,所以 $(ky - 1) \times (kx - 1)$ 也是已知量,故 $C_4$ 是已知量。在网络中,$ky$ 和 $kx$ 的大小通常相等,因此 $C_5 = ky - 1 = kx - 1$ 是已知量。因为 $C_4$ 和 $C_5$ 都是已知量,所以输入数据占用片上缓存空间大小由 $hp + wp$ 确定。根据基本不等式 $a + b \geqslant 2\sqrt{ab}$,当且仅当 $a$ 和 $b$ 相等时,$a + b$ 的结果取最小,所以当且仅当 $hp$ 和 $wp$ 相等时,$hp + wp$ 的值取最小。因为 $hp \times wp$ 和 $ho \times wo/M$ 相等,所以相同的输入数据规模在 PE 之间拆分时,占用总片上缓存空间大小有最小值 $M \times (2 \times C_4 \times \sqrt{ho \times wo/M} + C_5)$。

故结论是,当输入数据在 $HW$ 方向拆分时,应尽量保证每个 PE 处理的 $H$ 方向的大小和 $W$ 方向的大小相同。PE 的个数为 $M$,所以当 $ho/wo$ 接近 $M$ 时,只在 $H$ 方向拆分 $M$ 份;当 $ho/wo$ 接近 1 时,在 $H$ 方向拆分 $\sqrt{M}$ 份,在 $W$ 方向拆分 $\sqrt{M}$ 份;当 $ho/wo$ 接近 $1/M$ 时,只在 $W$ 方向拆分 $M$ 份。

## 5　实验和结果

本节将详细介绍本文实验所使用的 Benchmark 和实验平台,并将本文设计的 DCA 加速器与其他硬件平台进行性能对比。

### 5.1　Benchmark

本文的 Benchmark 选取了应用广泛的 MobileNet V1、MobileNet V2、MobileNet V3 网络中具有代表性的深度卷积规模作为实验测试用例,涵盖了神经网络中的各种深度卷积规模,如表 1 所示。

**表 1　深度卷积测试用例**

| 名称 | $hi$ | $wi$ | $c$ | $ky$, $kx$ | $sy$, $sx$ |
|---|---|---|---|---|---|
| dwconv1 | 114 | 114 | 16 | 3,3 | 1,1 |
| dwconv2 | 114 | 114 | 32 | 3,3 | 1,1 |
| dwconv3 | 114 | 114 | 64 | 3,3 | 2,2 |
| dwconv4 | 60 | 60 | 128 | 3,3 | 1,1 |
| dwconv5 | 30 | 30 | 256 | 3,3 | 1,1 |
| dwconv6 | 16 | 16 | 512 | 3,3 | 1,1 |
| dwconv7 | 9 | 9 | 1024 | 3,3 | 1,1 |

### 5.2　实验平台

本文在 3 种不同的平台上做了实验,分别是 CPU 平台、SIMD 平台和 DCA 加速器平台。

CPU 平台。通用多核处理器,Intel Core i7-8850H,基本频率是 2.60 GHz,最大睿频频率可达 4.30 GHz,9 MB L3 Cache,6 核,12 线程。算法的实现使用公开实现的高性能函数库,比如 Intel oneDNN[31]。

SIMD 平台。单核 SIMD 处理器。该平台的架构类似于 DCA 加速器平台,使用 0.75 MB 的 SPM,用于存放输入神经元、输出神经元和权值,时钟频率是 1 GHz,有运算器长度为 64 的乘法器和加法器。在 SIMD 平台,使用高性能库 DLPlib[25],数据摆放的格式是 *NHWC*。

DCA 加速器平台。本文提出的单核深度卷积加速器。0.25 MB 的 NBout,0.5 MB 的 NBin,时钟频率为 1 GHz。PE 个数设置为 16,向量 MAC 的长度为 16,因为根据本文 2.2 节的介绍,在实际应用的神经网络中 *C* 方向的最小值是 16。DCA 加速器使用的高性能库是在 DLPlib[25] 上实施本文第 4 节介绍的软件优化的版本。为了获得 DCA 加速器的性能,首先通过 DLPlib 生成指令,然后使用 C ++ 构建了一个性能模拟器来获取最终性能。为了与单核 SIMD 平台公平对比,DCA 加速器的访存带宽和单核 SIMD 处理器保持一致。

### 5.3 实验结果

软件优化。本文分析了软件优化对 DCA 加速器整体性能的影响。图 11 显示了使用软件优化的加速比,只使用权值单播模式,使用在此模式下最优的数据拆分方式。从图 11 中可以看出,在使用软件优化的情况下,所有的测试用例都有性能提升,平均提升了 1.65 倍。其中 dwconv1 的性能提升最明显,这主要是由于 dwconv1 的 *C* 方向比较小,使用 DCA 加速器的权值广播模式可以充分利用 DCA 的运算单元。

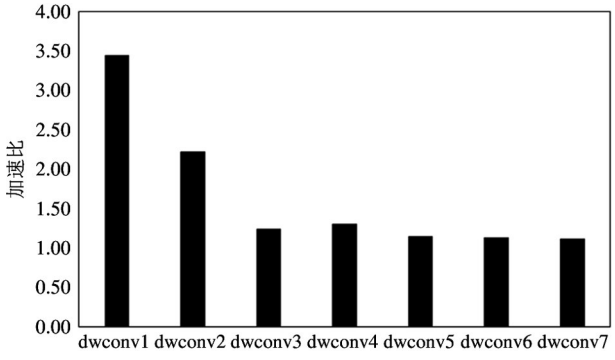


**图 11 DCA 加速器使用软件优化的加速比**

图 12 显示了表 1 的测试用例在不同的模式下的性能,其中每种模式都使用在当前模式下最优的拆分方式。从图 12 中可以看出,dwconv1 和 dwconv2 在权值单播模式下性能相对较差;dwconv3 和 dwconv4 在 3 种模式下性能相同;dwconv5、dwconv6 和 dwconv7 在权值单播模式下性能比较差。

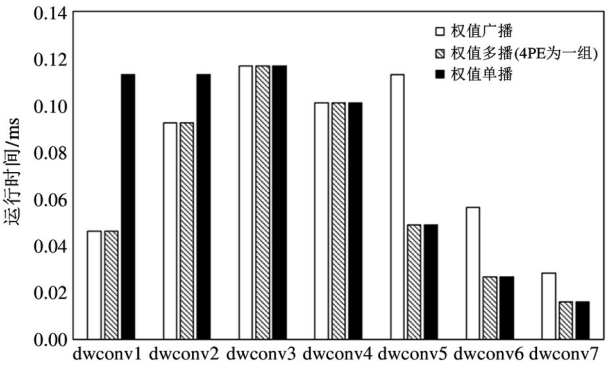


**图 12 测试用例在 DCA 加速器的不同模式下的运行时间对比**

在神经网络的前几层,*C* 方向通常比较小,如果使用权值单播模式,计算效率会非常低,最终导致计算成为整体性能的瓶颈。dwconv1 在权值广播模式下,访存时间大于计算时间。但是在使用权值单播模式下,它的计算效率只有 1/16,这导致了计算时间大于访存时间。在神经网络的中间几层,比如 dwconv3 和 dwconv4,由于其访存时间大于计算时间,虽然在使用权值广播和权值单播的模式时的计算效率降低,但是仍然是访存时间大于计算时间,所以在 3 种模式下性能一样。在神经网络的后几层,*C* 方向通常比较大,比如 dwconv5、dwconv6 和 dwconv7。如果选择使用权值广播模式,运算效率只有 1/16,只有一个 PE 在做有效工作,所以在权值广播模式下,这几层的计算是整体性能的瓶颈。总而言之,实验结果符合本文之前的理论分析。

不同平台对比。本文将 DCA 加速器的性能和通用 CPU、单核 SIMD 处理器进行对比。使用表 1 的深度卷积规模作为测试用例,测试结果如图 13 所示,其中横轴表示不同的深度卷积的规模,纵轴表示 DCA 加速器相对于其他平台的加速比。从图 13 可以看出,DCA 加速器与通用 CPU 处理器对比,平均加速了 7.9 倍,最大可加速 9.3 倍;DCA 加速器与单核 SIMD 处理器对比,平均加速了 8.3 倍,最大可加速 29.3 倍。

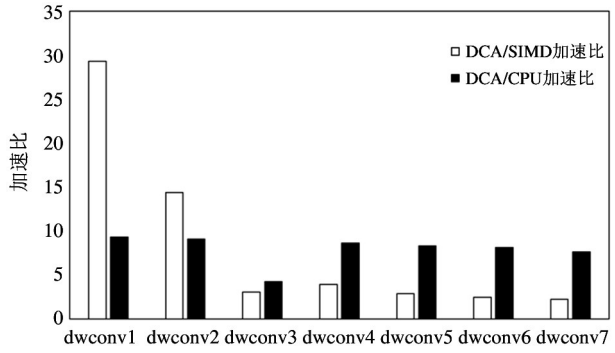


**图 13　DCA 加速器相对于 SIMD、CPU 平台的加速比**

DCA 处理器比通用 CPU 速度快的主要原因是 DCA 加速器的软件实现有数据预取操作,结合 DCA 加速器较大的算力,可以将计算时间完全隐藏在访存时间里。从图 13 可以看出,在 dwconv3 这个深度卷积规模下,DCA 加速器相对于 CPU 的加速比比其他规模的加速比小,因为 dwconv3 的 *stride* 大小是 2,即此时计算密度相对较小,这使得解决访存量决定性能的 DCA 加速器的优势降低。

DCA 加速器相对于 SIMD 的加速比最高是在 $C$ 方向值较小的深度卷积规模。因为固定向量运算器长度的单核 SIMD 处理器在这种情况下计算效率比较低,从而导致计算成为瓶颈,比如对于 dwconv1 的深度卷积规模,运算器长度为 64 的单核 SIMD 处理器的计算效率只有 25%。而 DCA 加速器在设计时考虑了包括较小 $C$ 的各种深度卷积规模。DCA 加速器通过软硬件协同优化,能够使得计算效率达到 100%,从而获得更好的整体性能。

## 6　结　论

许多加速器主要针对传统卷积算法进行优化,没有针对深度卷积进行深入分析和探讨。本文提出了一种软硬件结合的方法优化深度卷积,通过设计不同的权值数据传输模式,结合软件模式选择、数据拆分等优化方式,有效解决深度卷积中不同规模下的计算性能问题。实验结果表明,在实现 MobileNet 系列网络中的深度卷积时,与运算器长度为 64 的单核 SIMD 处理器相比,本文所描述的工作最高提升性能可达 29.3 倍。

未来将考虑对本文的 DCA 加速器做进一步扩展,以支持更多类型的卷积。DCA 加速器的多 PE、多种权值传输模式具有良好的扩展性,如果将每个 PE 的向量 MAC 运算器的结果分组累加或者全部累加,可以高效地计算权值规模相对较小的传统卷积,比如神经网络前几层的卷积。当权值很大时,可以将权值驻留在 NBin 上,传输输入,比如神经网络中后几层的卷积规模。

**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]// Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 2017: 4476-4484

[2] ABDEL-HAMID O, MOHAMED A R, JIANG H, et al. Convolutional neural networks for speech recognition[J]. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 2014, 22(10):1533-1545

[3] DOU Z Y, YU K Y, ANTONIOS A. Investigating meta-learning algorithms for low-resource natural language understanding tasks [EB/OL]. https:// arxiv. org/pdf/1908. 10423. pdf:arXiv, (2019-08-27),[2021-01-10]

[4] VENUGOPALAN S, ROHRBACH M, DONAHUE J, et al. Sequence to sequence-video to text[C]// Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 2015: 4534-4542

[5] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. *Communications of the ACM*, 2017, 60(6): 84-90

[6] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[EB/OL]. https:// arxiv. org/pdf/1409. 1556v6. pdf: arXiv, (2015-04-10),[2021-01-10]

[7] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognition [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016: 770-778

[8] SZEGEDY C, VANHOUCKE V, LOFFE S, et al. Rethinking the inception architecture for computer vision [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016: 2818-2826

[9] HAN S, MAO H Z, William J D. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding[EB/OL]. https://arxiv.org/pdf/1510.00149v5.pdf: arXiv, (2016-02-15), [2021-01-10]

[10] ZHOU S C, WU Y X, NI Z K, et al. DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients[EB/OL]. https://arxiv.org/pdf/1606.06160.pdf:arXiv, (2018-02-02),[2021-01-10]

[11] CHEN T S, DU Z D, SUN N H, et al. DianNao: a small-footprint high-throughput accelerator for ubiquitous machine learning[C]// Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, Salt Lake City, USA, 2014: 269-284

[12] CHEN Y J, LUO T, LIU S L, et al. DaDianNao: a machine learning supercomputer [C]// Proceedings of the IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014: 609-622

[13] Francois Chollet. Xception: deep learning with depthwise separable convolutions[EB/OL]. https://arxiv.org/pdf/1610.02357.pdf:arXiv, (2017-04-04),[2021-01-10]

[14] HOWARD A G, ZHU M, CHEN B, et al. Mobilenets: efficient convolutional neural networks for mobile vision applications [EB/OL]. https: // arxiv. org/pdf/1704.04861.pdf:arXiv, (2017-04-17),[2021-01-10]

[15] SANDLER M, HOWARD A, ZHU M, et al. Mobilenetv2: inverted residuals and linear bottlenecks [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 4510-4520

[16] HOWARD A, SANDLER M, CHU G, et al. Searching for mobilenetv3[EB/OL]. https://arxiv.org/pdf/1905.02244.pdf:arXiv, (2019-11-20),[2021-01-10]

[17] TAN M X, Quoc V Le. EfficientNet: rethinking model scaling for convolutional neural networks[EB/OL]. https://arxiv.org/pdf/1905.11946.pdf:arXiv, (2020-09-11),[2021-01-10]

[18] ZHANG P F, ERIC L, LU B T. High performance depthwise and pointwise convolutions on mobile devices[EB/OL]. https: // arxiv. org/pdf/2001.02504.pdf: arXiv, (2020-01-03),[2021-01-10]

[19] DU Z D, ROBERT F, CHEN T S, et al. ShiDianNao: shifting vision processing closer to the sensor[C]// Proceedings of the International Symposium on Computer Architecture, Portland, USA, 2015: 92-104

[20] 周聖元,杜子东,刘道福,等. 低面积低功耗的机器学习运算单元设计[J]. 高技术通讯, 2019,29(1):12-18

[21] 周聖元,杜子东,陈云霁. 稀疏神经网络加速器设计[J]. 高技术通讯, 2019,29(3):222-231

[22] ZHOU X D, DU Z D, GUO Q, et al. Cambricon-S: addressing irregularity in sparse neural networks through a cooperative software/hardware approach[C]// 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture, Fukuoka City, Japan, 2018:15-28

[23] LI Z, ZHI T, LIU E, et al. MW-DLA: a dynamic bit width deep learning accelerator[J]. *High Technology Letters*, 2020,26(2):145-151

[24] LIU S L, DU Z D, TAO J H, et al. Cambricon: an instruction set architecture for neural networks[C]// 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture, Seoul, Korea, 2016:393-405

[25] LAN H Y, WU L Y,WANG B R,et al. DLPlib: a library for deep learning processor [J]. *Journal of Computer Science and Technology*, 2017, 32(2) : 286-296

[26] DU W J, WU L Y, CHEN X B, et al. ZhuQue: a neural network programming model based on labeled data layout [C] // International Symposium on Advanced Parallel Processing Technologies, Tianjin, China, 2019: 27-39

[27] SONG J, ZHUANG Y M, CHEN X B, et al. Compiling optimization for neural network accelerators[C]// International Symposium on Advanced Parallel Processing Technologies, Tianjin, China, 2019:15-26

[28] CHEN T Q, MOREAU T, JIANG Z H, et al. TVM: an automated end-to-end optimizing compiler for deep learning[C]// The 13th USENIX Symposium on Operating Systems Design and Implementation, Carlsbad, USA, 2018: 578-594

[29] MENDIS C, BOSBOOM J, WU K, et al. Helium: lifting high-performance stencil kernels from stripped ×86 binaries to Halide DSL code[C]// Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, USA, 2015:391-402

[30] 陈云霁,李玲,李威,等. 智能计算系统[M]. 北京:机械工业出版社,2020:181-185

[31] Intel Corporation. oneAPI Deep Neural Network Library [EB/OL]. https://oneapi-src.github.io/oneDNN/index.html: Intel, (2020), [2021-01-10]

# Software and hardware co-optimization design and implementation of depthwise convolution

QI Hao*, LIU Shaoli**, LI Wei***

(*School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026)

(**Cambricon Technologies, Shanghai 201306)

(***State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Science, Beijing 100190)

**Abstract**

In recent years, deep learning technology has been widely used. As mobile devices are simultaneously limited by computing power and power consumption, many lightweight networks have been proposed, such as Xception, MobileNet series, etc. In these lightweight networks, the number of depthwise convolutional layers accounts for 31% -50% of all convolutional layers in the network, so how to optimize the operation of depthwise convolution is a problem worth studying. General-purpose CPUs and single instruction multiple data (SIMD) processors with fixed arithmetic unit length cannot efficiently process various scales of depthwise convolution in neural networks, and their performance is low. In response to this problem, this paper proposes a combination of software and hardware to optimize the calculation of depthwise convolution, through a hardware architecture design with multiple weight transmission modes combined with software mode selection and data splitting. This optimization method reduces the amount of memory access while improving computing efficiency. The experimental results show that the depthwise convolution accelerator implemented by this method can achieve a maximum performance acceleration of 9.3 times compared with a general-purpose CPU, and a maximum performance acceleration of 29.3 times compared with a single core SIMD processor with a length of 64.

**Key words**: neural network, depthwise convolution, accelerator, software and hardware collaborative optimization, computing efficiency