计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20170852
56(2):240-253, 2019

硬件加速神经网络综述

陈桂林 马 胜 郭 阳
(国防科技大学计算机学院 长沙 410073)
(cglnudt@163.com)

Survey on Accelerating Neural Network with Hardware

Chen Guilin, Ma Sheng, and Guo Yang
(*College of Computer*, *National University of Defense Technology*, *Changsha* 410073)

Abstract Artificial neural networks are widely used in artificial intelligence applications such as voice assistant, image recognition and natural language processing. With the rise of complexity of the application, the computational complexity has also increased dramatically. The traditional general-purpose processor is limited by the memory bandwidth and energy consumption when dealing with the complex neural network. People began to improve the architecture of the general-purpose processors to support the efficient processing of the neural network. In addition, the development of special-purpose accelerators becomes another way to accelerate processing of neural network. Compared with the general-purpose processor, it has lower energy consumption and higher performance. The article aims to introduce the designs from current general-purpose processors and special-purpose accelerators for supporting the neural network. It also summarizes the latest design innovation and breakthrough of the neural network acceleration platforms. In particular, the article provides an overview of the neural network and discusses the improvements made by various general-purpose chips to support neural networks, which include supporting low-precision operations and adding a calculation module to speed up neural network processing. Then from the viewpoint of the computational structure and storage structure, the article summarizes the customized designs of special-purpose accelerators, and describes the dataflow used by the neural network chips based on the reuse of various types of the data in the neural network. Through analyzing the advantages and disadvantages of these solutions, the article puts forward the future design trend and challenge of the neural network accelerator.

Key words machine learning; neural network; general-purpose processor; special-purpose accelerator; architecture

摘 要 人工神经网络目前广泛应用于人工智能的应用当中,如语音助手、图像识别和自然语言处理等.随着神经网络愈加复杂,计算量也急剧上升,传统的通用芯片在处理复杂神经网络时受到了带宽和能耗的限制,人们开始改进通用芯片的结构以支持神经网络的有效处理.此外,研发专用加速芯片也成为另一条加速神经网络处理的途径.与通用芯片相比,它能耗更低,性能更高.通过介绍目前通用芯片和专用芯片对神经网络所作的支持,了解最新神经网络硬件加速平台设计的创新点和突破口.具体来说,主要概述了神经网络的发展,讨论各类通用芯片为支持神经网络所作的改进,其中包括支持低精度运算和

收稿日期:2017-11-14;**修回日期**:2018-04-25
基金项目:国家自然科学基金项目(61672526);国防科技大学科研计划项目(ZK17-03-06)
This work was supported by the National Natural Science Foundation of China (61672526) and the Research Project of NUDT (ZK17-03-06).
通信作者:马胜(masheng@nudt.edu.cn)

增加一个加速神经网络处理的计算模块.然后从运算结构和存储结构的角度出发,归纳专用芯片在体系结构上所作的定制设计,另外根据神经网络中各类数据的重用总结了各个神经网络加速器所采用的数据流.最后通过对已有加速芯片的优缺点分析,给出了神经网络加速器未来的设计趋势和挑战.

关键词 机器学习;神经网络;通用芯片;专用加速芯片;体系结构

中图法分类号 TP303

随着数据的爆炸式增长和计算性能的阶跃式提升,机器学习的研究在经历20世纪由计算瓶颈引起的寒潮后重新变得火热起来.机器学习算法已广泛地运用到许多智能应用和云服务之中.常见应用有语音识别如苹果Siri和微软小娜,面部识别如Apple iPhoto或Google Picasa,同时智能机器人也大量使用了机器学习算法.目前,机器学习领域最火热的是以神经网络为代表的深度学习.据统计,深度神经网络运用在语音识别上比传统方法要减少了30%的单词识别错误率[1],将图像识别竞赛的错误率从2011年的26%降至3.5%[2-4].此外它还被广泛地应用到数据挖掘中.但由于神经网络的应用范围越来越广,精度要求越来越高,导致神经网络的规模也越来越大.通常对大规模神经网络加速的方法是设计性能更强大的通用芯片,并且增加专门的神经网络处理模块,如英特尔的Knight Mill和英伟达最新的Volte架构都添加了对神经网络的加速模块.不过它们作为通用运算器件始终限制了它们完成特殊任务时的效率,比如CPU必须包含高速缓存、分支预测、批处理、地址合并、多线程、上下文切换等多种通用功能,这些功能并不会完全用于神经网络的加速,但它们会占用芯片的设计面积.这就导致神经网络加速时硬件资源并没有完全利用.所以人们也开始设计专用芯片来实现对大型神经网络的加速.

近年来各大科研机构纷纷提出了各自的加速器结构,如中国科学院陈天石团队[5-8]的Diannao家族、Google的TPU[9](tensor process unit)和cloud TPU[10]、普度大学推出的Scaledeep[11]、MIT(Massachusetts Institute of Technology)提出的Eyeriss[12]、HP实验室和犹他大学联合提出的基于忆阻器的ISAAC[13],以及Parashar等人[14]提出的压缩稀疏卷积神经网络加速器SCNN等.现有神经网络加速芯片的研究主要集中在4个方面:1)从神经网络的计算结构出发,研究了树状结构和阵列结构如何高效地完成神经网络的卷积运算;2)从存储的瓶颈角度出发,研究了3D存储技术如何应用到加速器的设计中;3)从新材料器件的探索出发,研究了忆阻器等新器件如何实现神经网络处理存储一体化;4)从数据流的调度和优化出发,研究了如何最大化利用网络中各类数据的局部重用以及稀疏网络的处理.

1 神经网络概述

神经网络分为生物启发的神经网络(biologically inspired neural network)和人工神经网络(artificial neural network)2类.前者由神经生物学家关注,用来建立一种灵感来源于生物学的模型以帮助理解神经网络和大脑是如何运转的,比较有代表性的是脉冲神经网络.IBM的TrueNorth[15]和Furber等人[16-18]提出的SpiNNaker就是这种结构.但由于脉冲神经网络在处理机器学习任务时精度低的缺点,目前并没有得到广泛应用,本文不对其进行过多的赘述.后者人工神经网络是本文讨论的重点.

人工神经网络是一种典型的机器学习方法,也是深度学习的一种重要形式.1943年McCulloch和Pitts[19]提出了第1个人工神经元模型(M-P神经元如图1(a)),在这个模型中,神经元接受来自n个其他神经元传递过来的信号,这些输入信号通过带权重的连接进行传递,神经元接受到的总输入值将与神经元的阈值进行比较,然后通过"激活函数"产生神经元的输出.常用的激活函数有sigmoid函数、阶跃函数、ReLU等非线性函数.为了把人工神经网络研究从理论探讨付诸工程实践,1958年Rosenblatt[20]设计制作了感知机,如图1(b).感知机输入层并不是功能神经元,只接受外界输入信号后传递给输出层,输出层是一个M-P功能神经元.一个感知机可以实现逻辑与、或、非等简单的线性可分问题,但要解决一个复杂的非线性可分问题时,就要用到多层感知机(神经网络).如图1(c)的2层感知机可以解决一个异或问题,与单层感知机相比,它包含一个隐含层.当隐含层不断增加就有了深度神经网络(deep neural network, DNN)的概念,如图1(d).这里所谓的深度并没有严格的定义,在语音识别中4层网络就可以被认为是"较深的",而在图像识别中20层以上的网络比比皆是.不过随着

隐含层数的增加,深度神经网络有个明显的问题——参数数量的急剧膨胀.这就导致了计算量的急剧上升,进而使得网络模型的计算时间增加、计算功耗升高.为了解决这个问题,20 世纪 60 年代,Hubel 和 Wiesel[21] 提出了卷积神经网络(convolutional neural network, CNN)的概念.卷积神经网络局部感知和参数共享的特点使它能够有效地减少参数数量,降低深度神经网络的复杂性.

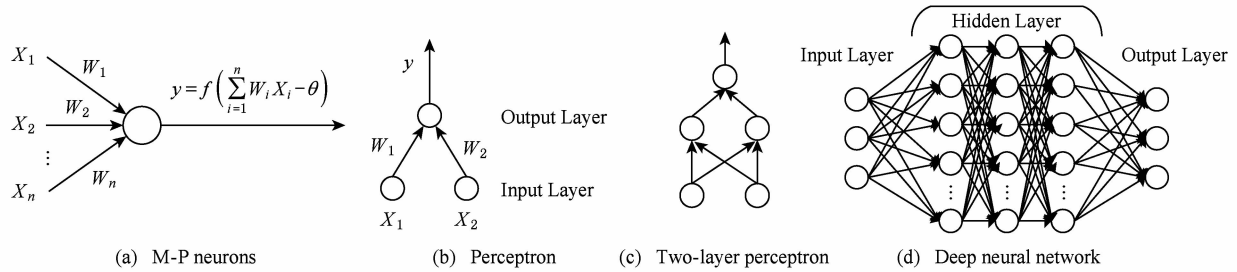

Fig. 1 The evolution of neural network

图 1 神经网络的演变过程

1) 局部感知,指通过对局部信息进行整合到达识别图像的目的.在传统的深度神经网络中,第 i 层每个神经元都会与第 $i-1$ 层所有神经元连接,这样做不仅导致了计算量非常大,而且网络缺少泛化能力,容易造成过拟合的情况.采用局部感知的方法,在机器进行图像识别时每个神经元不必对全局图像进行感知,只需要对局部进行感知,而后在更高层将局部的信息综合起来就得到了全局的信息.这样做不仅增强了网络的泛化能力,还有效减少了计算量.对应到具体操作是指上一层的数据区有一个滑动的窗口,只有窗口内的数据会与下一层神经元有关联,如图 2 所示.

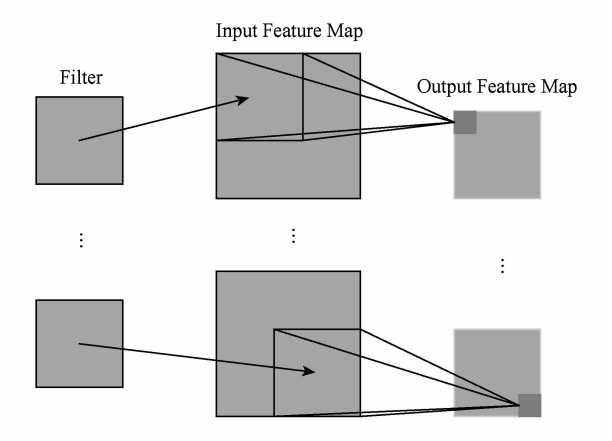

Fig. 2 The process of convolution

图 2 卷积的过程

2) 参数共享,即权值共享.对一个特征图来说,卷积操作就是提取特征的过程,这里简单介绍卷积核的概念,卷积核就是一个有相关权重的方阵,它的作用是用来提取图像的特定特征.提取特征和目标在图像中的位置无关,这也意味着某一部分学习的特征也能用在另一部分上,所以对于这个图像上的所有位置,都能使用同样的学习特征.具体来说就是将卷积核的每一个元素作用到输入特征图的每一个位置(边界因素先不考虑).参数共享保证了训练时对输入特征图只需要学习一个参数集合,而不是对每一个位置都需要学习一个单独的参数集合.

因为局部感知和参数共享这 2 个特点,卷积神经网络被广泛应用在图像识别领域.图 3 展示了

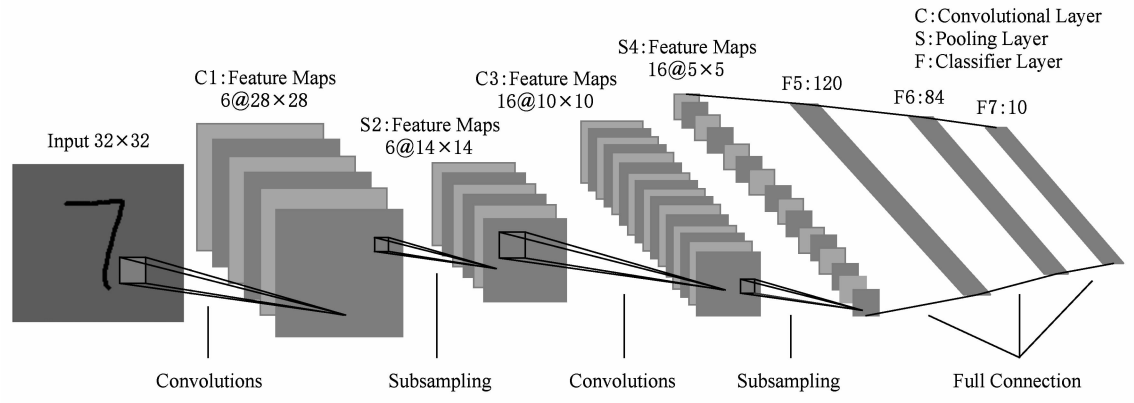

Fig. 3 A representative CNN architecture—LeNet5

图 3 一个具有代表性的 CNN 结构——LeNet5

一个经典卷积神经网络框架 LeNet5[22] 完成数字识别的过程.从图 3 中可以看出,卷积神经网络主要由卷积层、池化层和全连接层组成.

1.1 卷积层

介绍卷积层前,我们先介绍一下输入图像、输入特征图、输出特征图的概念.输入图像是指要进行识别的原始图像,并且它也是第 1 层网络的输入特征图,输出特征图是指神经网络某 1 层的输出,同时它也是下一层网络的输入特征图.卷积层是为了提取输入特征图中的某些特征,它的输出是由神经元组成的一幅新的 2D 特征图.计算过程是先卷积再通过一个激活函数得到结果,卷积过程如图 2 所示.计算公式为

$$out(x,y)=f\Big(\sum_{f_i=0}^{N_{f_i}}\Big(\sum_{k_x=0}^{K_x}\sum_{k_y=0}^{K_y}w_{f_i}(k_x,k_y)\times In(x+k_x,y+k_y)^{f_i}+\beta^{f_i}\Big)\Big),$$

其中,$f(\cdot)$是非线性激活函数,β^{f_i}表示偏移值,$out(x,y)$表示在输出特征图坐标(x,y)处的值,$w(k_x,k_y)$表示卷积核坐标(k_x,k_y)上的权重值,$In(x+k_x,y+k_y)$表示输入特征图坐标$(x+k_x,y+k_y)$上的输入值;K_x,K_y表示卷积核的大小,f_i表示第 i 幅输入特征图,N_{f_i}表示输入特征图的数目.

卷积层计算中卷积的实现方法有 4 种,直接卷积、展开式卷积(Toeplitz 矩阵方法[23])、Winograd 卷积[24]和快速傅里叶变换(fast Fourier transform,FFT)卷积[25].直接卷积方案在 Alex 编写的 cuda-convnet 框架[26]中有详细介绍.卷积过程如图 2 所示,与数字信号中的卷积运算不同[27],直接卷积是指卷积核在输入特征图上滑动时将滑动窗口内的元素对应相乘与累加(数字信号中的卷积会将卷积核翻转再与输入对应相乘累加).展开式卷积就是将输入图像展开成列,将卷积核展开成行,将卷积操作转换成矩阵乘法操作,最后通过高度优化数学库(GPU 的 cuBLAS)实现.Winograd 卷积是通过计算步骤的转换将卷积操作中的乘法操作和加法操作的数目改变,使乘法操作减少,加法操作增加,从而提升效率.FFT 卷积就是将空间域中的离散卷积转化为傅里叶域的乘积.它的实现分 3 个步骤:1)通过快速傅里叶变换将输入特征图和卷积核从空间域转换到频域;2)在频域中这些变换后的矩阵相乘;3)计算结果从频域反转到空间域.

1.2 池化层

池化层也叫下采样层,通常是跟在卷积层的后面,根据一定的采样规则(通常是最大值或平均值采样)[28]做采样.池化层的作用是提取局部特征.这是由于图像具有一种“静态性”的属性,在一个图像区域有用的特征可能在另一个区域同样适用.例如,卷积层输出的特征图中 2 个相邻的点的特征通常会很相似,假设 $a[0,0],a[0,1],a[1,0],a[1,1]$都表示颜色特征是红色,没有必要都保留作下一层的输入.池化层可以将这 4 个点做一个整合,输出红色这个特征.这样可以降低输入特征图的维度,减少过拟合现象,同时缩短网络模型的执行时间.

1.3 全连接层

全连接层的作用是将有用的局部信息提取整合,也就是将以前的局部特征通过权重矩阵重新组装成新的特征图.它的核心思想是非线性变换和拟合,多个全连接层的非线性变换嵌套叠加会使网络有很强的拟合能力(也可能造成过拟合).具体来说,在全连接层中,输出神经元与上层的输入神经元全部以独立的权重值相连接.我们可以将全连接层的计算看作是一个矩阵乘向量,再加上激活函数的非线性映射,即:

$$out_j=f\Big(\sum_{i=0}^{n-1}W_{ij}\times in_i+\beta\Big),$$

其中,out 是输出值,$f(\cdot)$是激活函数,n 表示输入特征图的输入数目,β代表偏移值.每个输入对应 1 个权重值.全连接层的运算实质也是一个乘累加运算.

以上就是一个卷积神经网络的基本组成,卷积神经网络的训练基本上和 BP(back propagation)神经网络的训练一样,通过递归不断地更新权重减小误差.现在又新兴的一种权重训练方法即采用遗传算法训练权重[29],文中总结的加速器大多只完成推导阶段,所以这里对训练不进行过多赘述.

2 通用芯片对人工神经网络的支持

常见的通用芯片包括 CPU,GPU,DSP,FPGA,随着深度学习的火热,它们的最新设计都对神经网络进行了一些支持.下面分别来介绍它们为支持神经网络所作的改进.

2.1 CPU

CPU 作为应用最广泛的通用处理器,其对神经网络的运算优化主要集中在软件层面,比如优化对神经网络框架(如 Caffe[30] 和 TensorFlow[31])的支持.硬件方面,随着研究表明推导时操作数的精度对准确率的影响不大,如表 1 所示:

Table 1 Effect of Operand Accuracy on Recognition Accuracy
表 1 操作数精度对识别准确率的影响

Type	Error Rate
32 b Floating-Point	0.031 1
16 b Floating-Point	0.033 7

CPU 也开始支持低精度运算. Intel 就对最新的处理器 Knights Mill[32] 增加了他们称之为“可变精度”的支持. 另外,它对向量处理单元(vector processing unit, VPU)也做了改进. 和上一代 Knights Landing 相比,Knights Mill 在 VPU 上用 1 个较小的双精度端口和 4 个向量神经网络指令(vector neural network instruction, VNNI)端口,取代了 Knights Landing 的 VPU 上 2 个大的双精度/单精度浮点(64 b/32 b)端口. VNNI 端口支持单精度浮点和混合精度整数(16 b 输入/32 b 输出). 由于操作数精度的降低,以往指令取出的 32 b 操作数如今变成了 2 个 16 b 操作数,这样运算相同数目的操作数将会采用更少的指令,同时也降低了计算的复杂度,进而提高了性能.

2.2 GPU

图形处理器 GPU 强大的并行计算能力适用于 CNN 计算过程中的大量并行浮点计算以及矩阵和向量运算,这促使了 GPU 成为最常见的神经网络硬件加速平台. 许多机器学习的框架都利用了含有 CUDA 编程接口的 GPU,如 cuda-convnet[33], Torch[34], Theano[35], Caffe[30]. 与此同时,许多 GPU 的深度学习库也在被探索用来加速神经网络,如英伟达的 cuDNN[36] 和 Facebook 的 fbfft[37] 等. 需要注意的是并没有哪一种框架或库对所有类型的神经网络都能有最好的加速效果. 针对不同类型、不同结构神经网络上述框架和库都有不同的表现.

除了支持深度学习库之外,GPU 也在体系结构上支持半精度的运算,如 tesla P100 采用的 GP100 核心、tesla V100 采用的 GV100 核心以及 AMD 的 Vega 架构. 英伟达最新的 Volta 架构更是设计了一个专门用来处理深度学习任务的 Tensor Core. 每个 Tensor Core 包含 1 个 4×4×4 的矩阵处理阵列来完成 $\boldsymbol{D}=\boldsymbol{A}\times\boldsymbol{B}+\boldsymbol{C}$ 的运算,其中 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}$ 是 4×4 的矩阵,如图 4 所示.

矩阵相乘的输入 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是 16 b 半精度浮点数(FP16)矩阵,矩阵 $\boldsymbol{C}$ 和 $\boldsymbol{D}$ 可能是 FP16 矩阵或 FP32 矩阵. 该 Tensor Core 支持 16 b 和 32 b 的混合精度运算. FP16 的乘法得到了一个全精度结果,该结果和上一周期得到部分和进行累加,如图 5 所示.

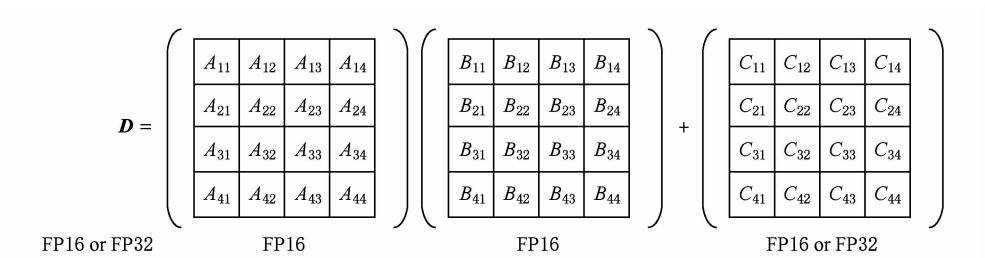

Fig. 4 4×4×4 matrix multiplication and accumulation of the Tensor Core

图 4 Tensor Core 的 4×4×4 矩阵乘法与累加

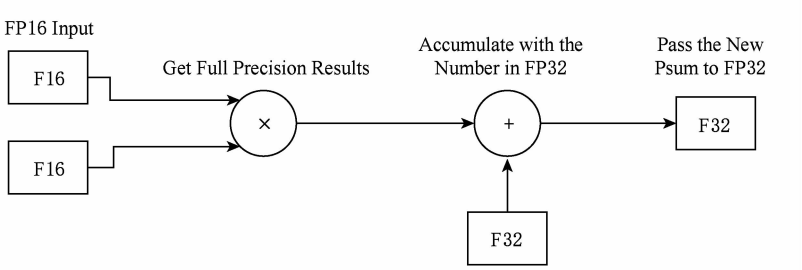

Fig. 5 Flow chart of the Tensor Core

图 5 Tensor Core 流程图

2.3 DSP

数字信号处理(DSP)芯片能够提供强大的数字信号处理能力,它也是一种实现神经网络加速的有效平台. 四大 DSP IP 产商也都发布了支持神经网络的 DSP IP,包括 Synopsys 的 EV6x(embedded vision)处理器[38]、CEVA 的 CEVA-XM6[39]、VeriSilicon 的 VIP8000[40] 以及 Cadence 的 Vision C5 DSP[41].

EV6x 在硬件结构上实现了对 CNN 的加速.

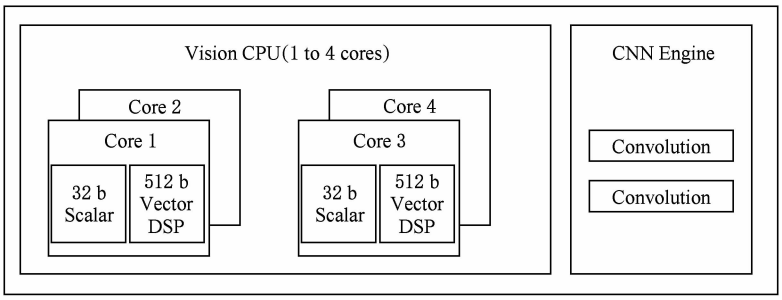

Fig. 6 The structure of EV6x

图 6 EV6x 结构图

为了增加对神经网络卷积运算的支持,从图 6 可以看出在 EV6x 的设计中引入了 1 个 CNN Engine. CNN Engine 是一个可编程嵌入式深层神经网络引擎,它支持当前所有主流神经网络模型(包括 AlexNet, GoogleNet, ResNet, VGG, Yolo, Faster R-CNN,SqueezeNet). CNN 引擎的可配置性使不同的网络模型可以灵活地映射到硬件资源. 它能提供每周期 800MAC(multiply-and-accumulate)(一般将 1 次 MAC 看作 2 个操作:1 个乘法和 1 个加法)性能,与同类解决方案相比,性能提高了 6 倍. 此外,神经网络还经常涉及到稀疏矩阵的处理,有效数据并没有连续存储,因此在 EV6x 的 4 个视觉 CPU 中提供了"分散-收集"(scatter-gather)功能,scatter-gather 是将不规则存放的数据一次取出,再组合成一个向量以便它们能够被 VPU 并行处理.

CEVE-XM6 针对神经网络处理做了 2 点优化:1)为了优化对神经网络中稀疏矩阵的处理,针对不规则访存增加了并行 Scatter-Gather 内存装载机制;2)采用了类似 EV6x 设计中 DSP 加硬件加速器的结构. 为了利用图像卷积处理时卷积核滑过输入特征图时出现的数据重叠,加速器中采用了滑动窗口 2.0(sliding windows 2.0)机制,这有助于在更广泛的神经网络中实现更高的利用率.

VIP8000 由高度多线程的并行处理单元、神经网络单元和通用存储缓存单元组成. 它具有可灵活配置的特点,可以导入由 Caffe 和 TensorFlow 等主流深度学习框架生成的神经网络. 它的并行处理单元、神经网络单元和通用存储单元都具有可扩展性. 为了实现最佳计算效率和准确率,神经网络单元支持定点 8 b 精度和 16 b 浮点精度,并支持混合模式应用. VIP8000 最终能取得每秒 3 TMAC 的性能.

最后,Cadence 的 Vision C5 DSP 为了实现神经网络所有层的计算(卷积层、池化层、全连接层),而不仅仅是卷积层的加速. 它将图像处理运算单元和神经网络加速单元合二为一. 通过移除神经网络加速器和主视觉/图像 DSP 之间的冗余数据传输,Vision C5 DSP 的功耗远低于现有的神经网络加速器. 为了支持低精度运算提高计算效率,它大量增加了低精度乘累加(MAC)单元的数量,有 1 024 个 8 b MAC 单元(512 个 16 b MAC 单元). 另外为了适应快速变换的神经网络领域,可编程的设计也是必不可少的.

通过比较以上 4 个 DSP IP 的设计,不难发现设计 DSP IP 实现神经网络硬件加速的一些共同点:1)都支持低精度的向量运算;2)具有可编程可扩展的特点;3)支持当前所有的主流神经网络框架.

2.4 FPGA

现场可编程门阵列(field programmable gate array,FPGA)也常被用作神经网络的加速平台,这类加速器利用可编程门阵列的特性,设计新的硬件结构,更好地匹配了神经网络的计算特点. 与 CPU 和 GPU 相比,它节省了部分通用功能所占芯片面积,更加高效地利用了硬件资源,性能更高并且能效也更高;与 ASIC(application specific integrated circuit)相比,FPGA 能够实现快速的硬件功能验证和评估,从而加快设计的迭代速度.

具体来说,FPGA 针对神经网络的优化主要包括 2 个方面:1)从计算结构的角度出发;2)从算法的角度出发. 计算结构上的优化尽量使神经网络计算并行化和流水化,并且利用 FPGA 可编程的特性来设计满足加速器的灵活性和可扩展性;算法上的优化主要针对卷积层,利用 FPGA 的可编程性硬件实现各类卷积方法. 下面通过几个例子介绍这些优化方法是如何实现的.

早在 2002 年 Yun 等人[42]就在 FPGA 上实现了多层感知机,他们提出了一种硬件实现数字神经网络的新体系结构——ERNA. 在传统 SIMD(single instruction multiple data)结构的基础上,采用灵活

的阶梯式总线和内部连接网络.所提出的架构实现了基于并行化和流水线化的快速处理,同时不放弃传统方法的灵活性和可扩展性.此外,用户还可以通过设置配置寄存器来改变网络拓扑.

2011 年 Farabet 等人[43]提出了基于 FPGA 的卷积神经网络处理器——NeuFlow.它是一种运行时可重构的数据流处理器,包含多个计算瓦片.在计算瓦片中,集成了多个 1D 卷积器(乘累加单元 MAC),形成 1 个 2D 卷积算子.输入输出块通过级联卷积器和其他具有可编程性的操作元件形成,然后再经过路由多路复用器连接到全局总线.

在 FPGA2017 会议上,张弛等人[44]提出了一种在 CPU-FPGA 共享内存系统上对卷积神经网络进行频域加速的方法.首先,利用快速傅里叶变换(FFT)和重叠加法(overlap-and-add, OaA)来减少卷积层的计算量.具体做法是将频域算法映射到 FPGA 上高度并行的基于 OaA 的 2D 卷积器上.然后在共享内存器中提出了一种新颖的数据布局,用于 CPU 和 FPGA 之间高效的数据通信.为了减少内存访问延迟并保持 FPGA 的峰值性能,该设计采用了双缓冲.为了减少层间数据重映射延迟,该设计利用了 CPU 和 FPGA 进行并发处理.通过使用 OaA,CNN 浮点运算次数可以减少 39.14%~54.10%.

总的来说,由于 FPGA 可编程性强,作为加速器研发周期短,在它上面实现神经网络加速的研究越来越多.但目前的深度神经网络(DNN)计算还是重度依赖密集的浮点矩阵乘法(GEMM),抛开独特的数据类型(利用稀疏压缩后的数据类型)设计,它更利于映射到 GPU 上(常规并行性).因此市场上依然是 GPU 被广泛地用于加速 DNN.FPGA 虽然提供了卓越的能耗效率,但它并不能达到当今 GPU 的性能.但是考虑到 FPGA 的技术进展以及 DNN 的算法创新速度,未来的高性能 FPGA 在处理 DNN 方面可能会优于 GPU 的性能.例如英特尔即将推出的 14 nm 的 Stratix,10 个 FPGA 将会具有数千个 DSP 和片上 RAM.还将具有高带宽存储器 HBM(一种 3D 存储技术).这种功能组合就提供了 FPGA 与 GPU 相差不多的浮点计算性能.再加上现在的 DNN 算法里开始利用稀疏(剪枝等产生)和紧凑的数据类型来提升算法的效率.这些自定义数据类型也引入了不规则的并行性,这对于 GPU 来说很难处理,但是利用 FPGA 的可定制性就能非常高效地解决.例如清华大学汪玉团队就在 FPGA 上优化了对神经网络稀疏性的处理.

以上 4 种通用硬件加速平台由于在优化神经网络处理的同时还要考虑其对通用计算的支持,因此并不能完全利用所有计算资源来完成神经网络的加速.此时,体系结构研究者考虑设计一种专用芯片来完成这一加速工作.

3 专用人工神经网络加速器

当前存在许多关于神经网络加速器的研究,本节从运算存储结构和数据流调度优化的角度对现有专用人工神经网络加速器设计进行分析梳理.

3.1 体系结构设计

现有加速器大都采用基于 CMOS(complementary metal oxide semiconductor)工艺的冯·诺依曼体系结构,这类加速器设计注重 2 个的模块:运算单元和存储单元.运算单元的实现分为 2 种:树状结构和阵列结构.存储单元的设计用来存储神经网络每一层的输入值、权重和输出值等参数.如何平衡片上片外的数据分配,最小化数据的搬移是它设计的重点.值得注意的是,随着器件工艺的发展,一些体系结构研究者开始采用忆阻器等新器件来设计处理存储一体化的加速器.这类加速器有效地解决了带宽的瓶颈,具有功耗低速度快的特点.另外,专用指令集也是体系结构设计的一大重点.下面分别介绍这几个方面.

3.1.1 CMOS 工艺下的运算单元结构

运算单元是加速器设计的重点,根据神经网络的计算特点,本文总结了 2 种运算单元的实现结构:1)树状结构,通过对神经元计算过程的抽象得到;2)PE 阵列结构,利用神经网络每一层有大量乘累加并行计算的特点,将乘累加操作放入 1 个 PE 里,这样可以通过 PE 阵列实现神经元的并行处理.具体设计方案如下.

图 7 给出了树状结构图.方框内是一个简易的神经元计算单元 NFU,最右端的根节点是每个神经元的输出,子节点包括乘法器、加法器和非线性激活函数,叶子节点是神经元的输入.特别地,图 7 中第 2 层的结果有的并没有经过 NFU-3 的函数激活,而是直接输出(图 7 中虚线部分),这是因为神经网络中有的层并没有激活操作,典型的如池化层.采用这种树状结构的设计有 DianNao[6],DaDianNao[5].两者都采用了 NFU 作为加速器的基本处理单元,不同的是 DaDianNao 有更好的扩展性,可以高性能地支持大尺寸网络模型.不过从 NFU 的结构可以看出 DianNao 和 DaDianNao 仅能很好地支持神经网络

的计算(NFU 只有乘累加运算单元).为了运行更多的机器学习算法,PuDianNao[7]设计了新的计算单元(machine learning unit, MLU),它的子节点包括计数器、加法器、乘法器、加法树、累加器(accumulator)和杂项作业(miscellaneous, Misc).支持了更多的机器学习算法,如 k-均值、k-最近邻、朴素贝叶斯、支持向量机、线性回归、神经网络、决策树等. PuDianNao 运行上述机器学习算法时的平均性能与主流通用图形处理器相当,但面积和功耗仅为后者的百分之一量级.

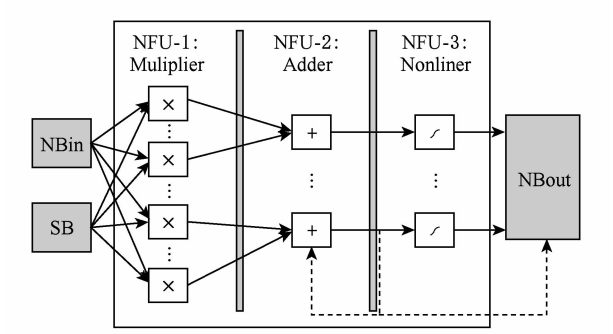

Fig. 7 A typical tree structure of NFU

图 7 一个典型树状结构的 NFU

阵列结构主要以 Google TPU[9]的脉动阵列结构为代表. TETRIS[45], Eyeriss[12], ShiDianNao[8], Scaledeep[11]所采用的处理结构都在脉动阵列上进行或多或少的改动.脉动阵列的设计核心是让数据在运算单元的阵列中进行流动,增加数据的复用,减少访存的次数.图 8 展示了 TPU 中矩阵乘法单元的脉动数据流.权重由上向下流动,输入特征图的数据从左向右流动.在最下方有一些累加单元,主要用于权重矩阵或者输入特征图超出矩阵运算单元范围

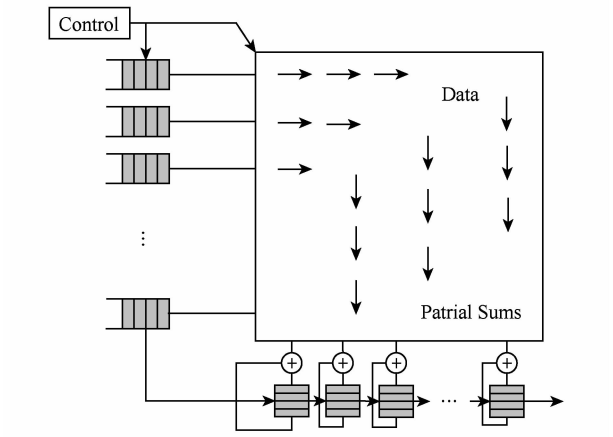

Fig. 8 Systolic data flow of the matrix multiply unit in TPU

图 8 TPU 矩阵运算单元的脉动数据流

时保存部分结果.控制单元负责数据的组织,具体来说就是控制权重和输入特征图的数据如何传入脉动阵列以及如何在脉动阵列中进行处理和流动.

3.1.2 CMOS 工艺下的存储结构

因为数据存取的速度大大低于数据处理的速度,因此存储单元的设计直接影响到加速器的性能.英伟达公司首席科学家 Steve[46]曾指出,在 40 nm 工艺下,将 64 b 数据搬运 20 mm 的能耗是做 64 b 浮点乘法的数倍.因此,要降低处理器功耗,仅仅优化处理结构是不够的,必须对片上存储单元的结构也进行优化.传统的存储单元设计是将不同数据放在同一个存储器里.在 DianNao 里提出了一种存储方式(图 7),对神经网络参数进行分块存储(用于存放输入神经元的输入缓冲器 NBin、存放输出神经元的输出缓冲器 NBout、存放突触权重的缓冲器 SB),将不同类型的数据块存放在片上不同的随机存储器中,再优化神经网络运算过程中对不同类型数据的调度策略.与 CPU/GPU 上基于缓存层次的数据搬运方法相比,DianNao 的设计方案可将数据搬运次数减少至前者的 1/30~1/10. DaDianNao 也延续了这种存储结构设计,同时 DaDianNao 使用了 eDRAM 来存放所有的权重值并将其放在计算部件附近以减少访存延时.

另外,随着神经网络规模越来越大,训练集越来越大,处理能力也越来越强,对带宽的要求就变得越来越高.传统 2D 存储结构的 DDR 技术不能提供如此高的带宽.人们开始将 3D 存储技术引入到神经网络加速器的设计中.现在 3D 存储方案有 AMD 和海力士研发的 HBM 以及 Intel 和美光研发的 HMC. Google 的 Cloud TPU[9]采用了 HBM 作为存储结构. Neurocube[47]和 TETRIS[45]采用了 HMC 作为加速器的存储结构.另外,3D 存储的出现使人们开始考虑将累加器设计到 3D 存储体里.这样做可以减少 1 次数据的读取,从而降低延迟,节省功耗,提高性能. TETRIS 给出了 4 种在 HMC 中设计累加器的方案,分别是内存控制器级累加、DRAM 芯片级累加、Bank 级累加和子阵列级累加.第 1 种方案是将累加器做在 HMC 的逻辑层,这样设计并不能减少数据的读写次数.后 3 种设计能带来性能提升,但子阵列级累加实现困难,所以主要还是采用 DRAM 级累加和 Bank 级累加.

3.1.3 新工艺下的一体化结构

运算存储一体化结构的加速器打破了冯·诺依曼体系结构的束缚,要想实现这类加速器依靠传统

CMOS 工艺较为困难,新型器件的研究和非传统电路的实现是这类加速器的研究方向.最新研究表明忆阻器是实现一体化结构的一种较好选择,它本身具有存储数据的功能,另外利用基尔霍夫定律产生的位线电流就是卷积运算乘累加的结果,节省了乘法和累加的计算时间.去年,在忆阻器领域具有领先地位的 HP 公司与犹他大学合作研究发表了基于忆阻器的神经网络加速器 ISAAC[13].它的结构与 DaDianNao[5]相似,由多个 Nodes 或 Tiles 组成,忆阻器交叉(crossbar)开关阵列组成了每个 Tile 的核心来完成存储权重和卷积计算的功能.与 DaDianNao 相比,吞吐量是其 14.8 倍,功效是其 5.5 倍,计算密度是其 7.5 倍.

目前做神经网络加速的忆阻器器件主要以 ReRAM 为主.加州大学圣塔芭芭拉分校谢源教授课题组也发布了一种基于 ReRAM 的神经网络加速器 PRIME[48].基于新型材料的 ReRAM 被认为是今后可以替代当前 DRAM 的下一代存储技术之一.作为忆阻器的一种,ReRAM 除了是存储单元之外,其独特的交叉网络结构(crossbar)和多比特存储(multi-level cell)性质,能以很高的能量效率加速神经网络计算中的重要计算模块——乘累加.实现方法是通过在 ReRAM 存储体内修改一部分外围电路,使其可以在“存储”状态和“神经网络加速器”状态之间灵活切换.

现阶段使用 ReRAM 之类的忆阻器器件来实现 CNN 加速器还有许多挑战,如精确度、内部数据调度以及模数转换等.清华大学的张悠慧教授等人[49]提出了神经网络模型转换方法,将原神经网络稀疏化后划分成规模适应于 ReRAM 阵列的子网络,并对数据进行量化来解决硬件精度受限问题,这一定程度上从软件的角度解决了计算精度的问题.而李楚曦等人[50]提出的基于忆阻器的 PIM 结构实现深度卷积神经网络近似计算,通过利用模拟忆阻器大大增加数据密度,并将卷积过程分解到不同形式的忆阻器阵列中分别计算,增加了数据并行性,减少了数据转换次数并消除了中间存储,从而实现了加速和节能.但是目前实现人工神经网络大多还是采用存储分离的结构,不过随着忆阻器技术的不断成熟,由于其低功耗和快速处理数据的特点,新工艺下的一体化结构终将会是加速器发展的一大趋势.

3.1.4 指令集设计

与基于 RISC 指令集的通用处理器不同,专用芯片多采用基于 CISC 来设计自己指令集.这类指令集侧重于完成更复杂的任务,适用于指令数量少和复杂程度高的神经网络.传统指令集实现一个神经元的操作需要上百条指令,如果使用专有指令集一个神经元的操作可以只用一条指令执行,减少功耗的同时提升了速度.TPU 的指令集就是 CISC 类型,平均每个指令的时钟周期数是 10~20.指令总共只有十几条.其中比较重要的包括读权重值和读输入特征图数据、矩阵运算/卷积、激活和写回计算结果等一系列神经网络运算相关的指令.

不过另一方面.越来越多研究者指出指令集的设计除了专用性外,还需要足够的灵活来满足加速器处理不同的神经网络的效率.为了解决这个问题,中科院陈天石团队[51]基于 RISC 设计了一种新的用于神经网络加速器领域的专用指令集架构,称为 Cambricon,它是一种集标量、向量、逻辑、数据传输和控制指令于一体的存取架构.Cambricon 在广泛的神经网络技术上表现出强大的描述能力,并且比通用指令集架构(X86,MIPS,GPGPU)提供更高的代码密度.

3.2 数据流调度和优化

3.1 节介绍了专用芯片在体系结构方面对神经网络所作的支持,但是在神经网络处理过程中数据流的组织和调度的方式以及数据流的优化对硬件实现神经网络的性能影响同样很大.在神经网络中常用的数据流有权重固定流、输出固定流、无局部复用和行固定流,常用的数据流优化处理手段有 0 值跳过、稀疏矩阵、参数压缩等.

3.2.1 数据流的调度

1) 权重固定流

在卷积神经网络的卷积操作过程中,同一个卷积核会卷积多个输入特征图.此时对这个卷积核里的权重是有复用的.如果把多次使用的权重存在每个 PE 的寄存器里,那么卷积层的计算过程将会减少从全局缓冲取数据和向全局缓冲存数据的开销.由于部分和(partial sum, Psum)还会用到,因此将它暂时存放在全局缓冲里,如果缓冲区不大就必须限制同时产生 Psum 的数量,这样也限制了一次性可以在片上加载的权重.此时若采用脉冲的方式将会有效减少 Psum 对全局缓冲大小的依赖,图 9(a)给出了这种数据流的处理过程.权重值存放在 PE 里,输入激活值被广播到所有 PE 单元.部分和在每个 PE 累加过后传到下一个 PE.

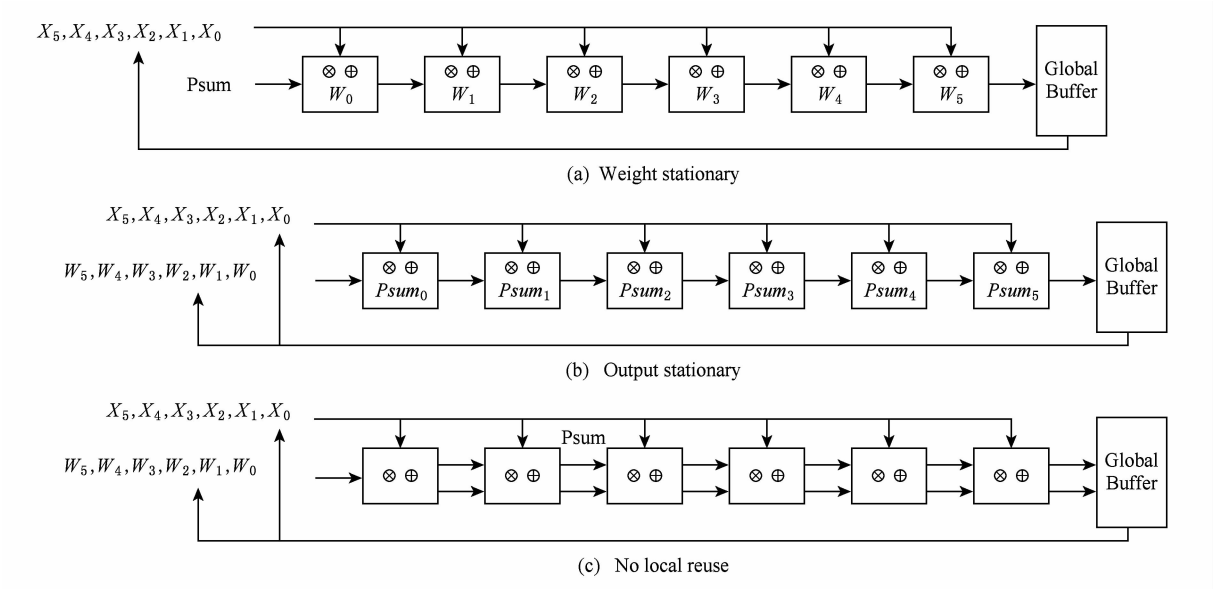

Fig. 9 Dataflow of the neural network

图 9 神经网络的数据流

采用这种数据流的芯片设计有 NneuFlow[43],它用了 2D 卷积引擎来处理卷积核.每个引擎处理过程中权重值保持不变.其他采用这种数据流的例子还有[52-54].

2) 输出固定流

指在寄存器中存放每个周期计算的部分和.卷积操作是一个乘累加操作,每个周期计算的部分和需要与下一个周期的部分和相加得到新的部分和,直到这个卷积核的计算完全结束.把部分和存在寄存器里同样也是减少了从全局缓冲取数据和向全局缓冲存数据的开销.同样这种数据流设计采用脉冲结构实现是最佳的.图 9(b)给出了这种数据流的处理过程.部分和在 PE 内部累积最后流出到全局缓冲区,权重值被广播到所有 PE.输入激活值在 PE 进行乘法运算后传给下一个 PE.

采用这种数据流的芯片设计有 ShiDianNao[8].它的每个 PE 通过从相邻 PE 获取相应的输入激活来处理每个输出激活值.PE 阵列通过实现专用网络来水平和垂直地传递数据.每个 PE 还具有数据延迟寄存器,以便在所需的循环周期内保持数据周期.全局缓冲区流入输入激活值,并将权重广播到 PE 阵列中.部分和积累在每个 PE 内部,然后被流出回到全局缓冲区.其他采用这种数据流的例子还有[55-56].

3) 无局部复用

指不在 PE 的寄存器文件里存放权重和部分和,每个周期需要计算的数据都从全局缓冲获取,计算完后将结果传回全局缓冲,最后通过 PE 阵列累积部分和.这样的好处是减少了 PE 所占的面积,但是由于 PE 没有数据保持固定,所以这样的设计也增加了全局缓冲区和空间 PE 阵列的流量.图 9(c)给出了这类数据流的处理过程.

采用这种数据流的芯片设计有 UCLA[57],DianNao[6],DaDianNao[5],PuDianNao[7].其中 DianNao 系列由于采用了专门的寄存器来保存部分和,并不用从内存里取,因此可以进一步降低访问部分和的功耗.

4) 行固定流

指在高维卷积过程中通过将高维卷积分解成可以并行运行的 1D 卷积原语;每个基元(原语操作)表示 1 行卷积核权重和 1 行输入特征图像素的点积,并生成 1 行部分和 Psum.不同原语的 Psum 被进一步累积在一起以生成输出特征图.具体做法是将每个原语映射到 1 个 PE 进行处理,每个行对的计算在 PE 中保持独立,这就实现了卷积核和寄存器级别的输入特征图数据重用.图 10 给出了这类数据流的处理过程.

采用这种数据流的芯片设计有 Eyeriss[12] 和 TETRIS[45].

5) 其他数据流

上述数据流都没有将权重值和输入激活值(上一层输出经过激活函数得到的值)完全复用.在 SCNN[14] 中提出了一种全新的数据流,笛卡儿数据流.实现方法是将需要计算的权重和输入激活值做成 2 个向量,然后 2 个向量做笛卡儿积,这样向量中每个数能达到最大复用.

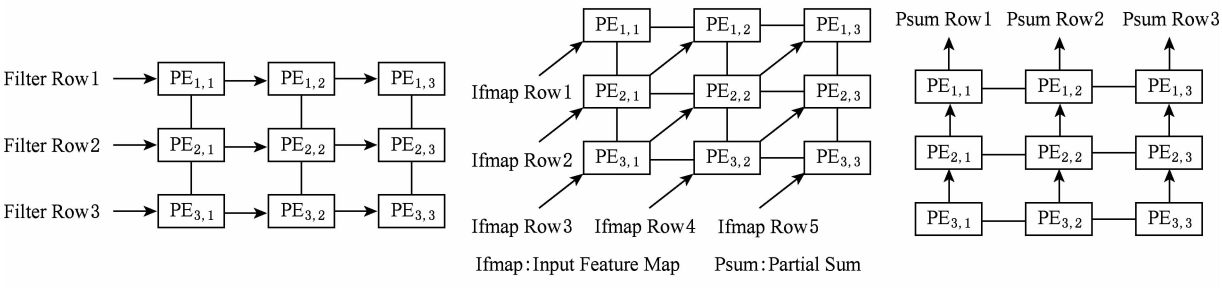

Fig. 10 The row stationary dataflow in PEs set to process a 2D convolution

图 10 处理 2 维卷积时的 PE 阵列中的行固定流示意图

3.2.2 数据流的优化

神经网络在数据的处理过程中还存在很多的优化技巧. 首先,神经网络在推导过程中的对精度要求不高,这使几乎全部的神经网络专用加速器都支持低精度运算,其中 TPU[9] 更是支持了 8 b 低精度运算. 其次,稀疏性. 人们在神经网络训练和推导过程中发现会产生许多的 0 值,一部分来源于网络训练阶段的剪枝,另外一部分来源于推导阶段采用 ReLU 激活函数参数的 0 激活值. 而 0 值在卷积操作中是可以跳过的,因为卷积操作由多个乘累加操作组成,乘数有一个为 0 则乘法和后面的加法就没必要计算. 在 Cnvlutin[58] 就采用了跳过权重中 0 值的数据流,在 Cambricon-X[59] 里则采用了跳过输入的 0 激活值,而在 SCNN 里通过压缩稀疏性和将非 0 参数编码的方式同时做到了跳过 0 值权重和 0 激活值. 这也引出了神经网络的另一种优化方法——压缩. 在 SCNN[14] 和 EIE[60] 中都采用了压缩参数的方法.

虽说压缩稀疏权重从计算量的角度来看能够直接提高性能,但最近的研究表明在不损失精度的情况下直接通过权重剪枝却会造成性能的下降[56],这是由于解码压缩的权重需要额外的时间,并且压缩的非 0 权重值需要额外的索引值来记录它们原来矩阵的位置,这也增加了存储开销. 最终的结果导致只有在大量剪枝的情况下,剪枝后的新网络的性能才会优于剪枝前的网络,而且会有精度的损失. 在文献[61]中提出了 2 种解决办法分别针对并行性不同的 2 种硬件:1)采用基于 SIMD 的权重剪枝用于低并行度的硬件,实现方法是采用权重分组索引,这样一条 SIMD 指令可以读取多个权重值;2)使用节点剪枝用于高并行的硬件,具体实现思想是利用正则化既不完全去掉参数又可以将它的影响降到最小.

4 加速器未来发展的挑战和机遇

第 2 节和 3 节论述了许多硬件加速神经网络的方案,其中在通用芯片平台实现的包括支持低精度计算、支持更多的神经网络框架以及设计一个加速卷积运算的模块;在专用芯片平台实现的包括改进运算存储结构、优化数据流和设计专用指令集. 虽然这些设计已经在神经网络加速方面取得了重大进展. 但还是有许多问题和挑战要解决. 下面列举的 4 个方面或许会是加速器未来研究的可行方向.

1) 设计功耗更低的加速器. 嵌入式应用是加速器应用的一种趋势,加速器未来可能会应用到像可穿戴这样的领域,这就需要把功耗进一步降低,可突破的方向包括采用新器件,进一步探索数据的组织形式以减少数据的移动等.

2) 设计通用性更强的加速器. 随着神经网络应用的增多,神经网络的结构和框架也会越来越多. 未来加速器设计需要考虑到支持各个框架的核心算法.

3) 解决访存瓶颈. 存取速度跟不上运算速度依旧是加速器设计的难题. 目前 3D 存储是解决该问题的一个方向. 不过随着新工艺的发展,采用新器件的非冯•诺依曼体系结构或许能进一步改善这个问题.

4) 应用其他领域的突破成果. 技术革命通常伴随着不同领域的飞跃,采用基于生物启发的脉冲神经网络、量子计算机以及使用忆阻器等新器件都可能是加速器未来设计的可行方案.

参 考 文 献

[1] Dean J. Large-scale deep learning for building intelligent computer systems [C] //Proc of the 9th ACM Int Conf on Web Search and Data Mining. New York: ACM, 2016: 1-1

[2] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [C] //Proc of the 26th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2012: 1097-1105

[3] Szegedy C, Liu Wei, Jia Yangqing, et al. Going deeper with convolutions [C] //Proc of the 32nd IEEE Conf on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society, 2015: 1-9

[4] He Kaiming, Zhang Xianyu, Ren Shaoqing, et al. Identity mappings in deep residual networks [C] //Proc of the 2016 European Conf on Computer Vision. Berlin: Springer, 2016: 630-645

[5] Chen Yunji, Luo Tao, Liu Shaoli, et al. DaDianNao: A machine-learning supercomputer [C] //Proc of the 47th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2014: 609-622

[6] Chen Yunji, Chen Tianshi, Du Zidong, et al. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning [C] //Proc of the 19th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2014: 269-284

[7] Liu Daofu, Chen Tianshi, Liu Shaoli, et al. PuDianNao: A polyvalent machine learning accelerator [C] //Proc of the 20th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2015: 369-381

[8] Du Zidong, Fasthuber R, Chen Tianshi, et al. ShiDianNao: Shifting vision processing closer to the sensor [C] //Proc of the 42nd Annual Int Symp on Computer Architecture. New York: ACM, 2015: 92-104

[9] Jouppi N P, Young C, Patil N, et al. In-Datacenter performance analysis of a tensor processing unit [C] //Proc of the 44th Annual Int Symp on Computer Architecture . New York: ACM, 2017: 1-12

[10] Google. Cloud TPUs: Google's second-generation tensor processing unit is coming to cloud [EB/OL]. [2017-10-30]. https://ai. google/tools/cloud-tpus/

[11] Venkataramani S, Dubey P, Raghunathan A, et al. ScaleDeep: A scalable compute architecture for learning and evaluating deep networks [C] //Proc of the 44th Annual Int Symp on Computer Architecture. New York: ACM, 2017: 13-26

[12] Chen Yu-Hsin, Emer J, Sze V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks [C] //Proc of the 43rd Annual Int Symp on Computer Architecture. New York: ACM, 2016: 367-379

[13] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars [C] //Proc of the 43rd Annual Int Symp on Computer Architecture. New York: ACM, 2016: 14-26

[14] Parashar A, Rhu M, Mukkara A, et al. SCNN: An accelerator for compressed-sparse convolutional neural networks [C] //Proc of the 44th Annual Int Symp on Computer Architecture. New York: ACM, 2017: 27-40

[15] Akopyan F, Sawada J, Cassidy A, et al. TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34 (10): 1537-1557

[16] Furber S B, Lester D R, Plana L A, et al. Overview of the spiNNaker system architecture [J]. IEEE Transactions on Computers, 2013, 62(12): 2454-2467

[17] Furber S B, Galluppi F, Temple S, et al. The SpiNNaker project [J]. Proceedings of the IEEE, 2014, 102(5): 652-665

[18] Jin Xin, Lujan M, Plana L A, et al. Modeling spiking neural networks on SpiNNaker [J]. Computing in Science & Engineering, 2010, 12(5): 91-97

[19] McCulloch W S, Pitts W. A logical calculus of the ideas immanent in nervous activity [J]. Bulletin of Mathematical Biophysics, 1943, 5(4): 115-133

[20] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain [J]. Psychological Review, 1958, 65(6): 386-408

[21] Hubel D H, Wiesel T N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex [J]. Journal of Physiology, 1962, 160(1): 106-154

[22] 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

[23] Böttcher A, Silbermann B. Introduction to Large Truncated Toeplitz Matrices [M]. Berlin: Springer, 1999

[24] Winograd S. Arithmetic Complexity of Computations [M]. Philiadelphia PA: Society for Industrial & Applied Mathematics, 1980

[25] Vasilache N, Johnson J, Mathieu M, et al. Fast convolutional nets with fbfft: A GPU performance evaluation [OL]. [2018-04-13]. https://arxiv. org/abs/1412. 7580

[26] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [C] // Proc of the 26th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT Press 2012: 1097-1105

[27] Smith S W. The Scientist and Engineer's Guide to Digital Signal Processing [M]. Poway, CA: California Technical Publishing, 1997

[28] He Kaiming, Zhang Xianyu, Ren Shaoqing, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2014, 37(9): 1904-1916

[29] Dong L T, Mintram R. Genetic algorithm-neural network (GANN): A study of neural network activation functions and depth of genetic algorithm search applied to feature selection [J]. International Journal of Machine Learning & Cybernetics, 2010, 1(1/2/3/4): 75-87

[30] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding [C] //Proc of the 22nd ACM Int Conf on Multimedia. New York: ACM, 2014: 675-678

[31] Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed Systems [OL]. [2018-04-13]. https://arxiv.org/abs/1603.04467

[32] Patrick K. Intel Xeon Phi Knights Mill for machine learning [EB/OL]. (2017-08-21) [2017-10-18]. https://www.servethehome.com/intel-knights-mill-for-machine-learning/

[33] Anker G. Cuda-convnet2: A fast C++/Cuda implementation of convolutional (or more generally, feedforward) neural networks [EB/OL]. [2017-10-20]. https://code.google.com/p/cuda-convnet2/

[34] Collobert R, Kavukcuoglu K, Farabet C. Torch7: A matlab-like environment for machine learning [C/OL] //Proc of NIPS Workshop. 2011[2018-04-13]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.231.4195

[35] Bastien F, Lamblin P, Pascanu R, et al. Theano: New features and speed improvements [J/OL]. Computer Science, 2012 [2018-04-13]. https://arxiv.org/abs/1211.5590

[36] NVIDIA. Cudnn: GPU-accelerated library of primitives for deep neural networks [EB/OL]. [2017-10-30]. https://developer.nvidia.com/cuDNN

[37] Vasilache N, Johnson J, Mathieu M, et al. Fast convolutional nets with fbfft: A GPU performance evaluation [OL]. [2018-04-13]. https://arxiv.org/abs/1412.7580

[38] Cooper G. Facial expression analysis with deep learning & computer vision [EB/OL]. [2018-04-13]. https://www.synopsys.com/designware-ip/technical-bulletin/ev-facial-expression-dwtb-q117.html

[39] CEVA. CEVA-XM6: Fifth-generation computer vision and deep learning embedded platform [EB/OL]. [2017-10-30]]. https://www.ceva-dsp.com/product/ceva-xm6/

[40] Miya K. VeriSilicon's Vivante VIP8000 neural network processor IP delivers over 3 Tera MACs Per second [EB/OL]. [2017-10-30]. http://www.verisilicon.com/newsdetail_499_VivanteVIP8000.html

[41] Cadence. Tensilica Vision DSPs for imaging, computer vision, and neural networks [EB/OL]. [2017-10-18]. https://ip.cadence.com/vision

[42] Yun S B, Kim Y J, Dong S S, et al. Hardware implementation of neural network with expansible and reconfigurable architecture [C] //Proc of the 9th Int Conf on Neural Information Processing. Piscataway, NJ: IEEE, 2002: 970-975

[43] Farabet C, Martini B, Corda B, et al. NeuFlow: A runtime reconfigurable dataflow processor for vision [C] //Proc of the 29th Computer Vision and Pattern Recognition Workshops. Piscataway, NJ: IEEE, 2011: 109-116

[44] Zhang Chi, Prasanna V. Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system [C] //Proc of the 25th ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2017: 35-44

[45] Gao Mingyu, Pu Jing, Yang Xuan, et al. TETRIS: Scalable and efficient neural network acceleration with 3D memory [C] //Proc of the 22nd Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2017: 751-764

[46] Li Zhen, Wang Yuqing, Zhi Tian, et al. A survey of neural network accelerators [J]. Frontiers of Computer Science, 2017, 11(5): 746-761

[47] Kim D, Kung J, Chai S, et al. Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory [C] //Proc of the 43rd Annual Int Symp on Computer Architecture. New York: ACM, 2016: 380-392

[48] Chi Ping, Li Shuangchen, Xu Cong, et al. PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory [C] //Proc of the 44th Annual Int Symp on Computer Architecture. New York: ACM, 2016: 27-39

[49] Ji Yu, Zhang Youhui, Li Shuangchen, et al. NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints [C] //Proc of the 49th IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2016: No. 21

[50] Li Chuxi, Fan Xiaoya, Zhao Changhe, et al. A memristor-based processing-in-memory architecture for deep convolutional neural networks approximate computation [J]. Journal of Computer Research and Development, 2017, 54(6): 1367-1380 (in Chinese)
(李楚曦, 樊晓桠, 赵昌和, 等. 基于忆阻器的PIM结构实现深度卷积神经网络近似计算[J]. 计算机研究与发展, 2017, 54(6): 1367-1380)

[51] Liu Shaoli, Du Zidong, Tao Jinhua, et al. Cambricon: An instruction set architecture for neural networks [C] //Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 393-405

[52] Sankaradas M, Jakkula V, Cadambi S, et al. A massively parallel coprocessor for convolutional neural networks [C] //Proc of the 20th IEEE Int Conf on Application-Specific Systems, Architectures and Processors. Piscataway, NJ: IEEE, 2009: 53-60

[53] Sriram V, Cox D, Tsoi K H, et al. Towards an embedded biologically-inspired machine vision processor [C] //Proc of the 9th Int Conf on Field-Programmable Technology. Piscataway, NJ: IEEE, 2011: 273-278

[54] Chakradhar S, Sankaradas M, Jakkula V, et al. A dynamically configurable coprocessor for convolutional neural networks [C] //Proc of the 38th Int Symp on Computer Architecture. New York: ACM, 2010: 247-257

[55] Gupta S, Agrawal A, Gopalakrishnan K, et al. Deep learning with limited numerical precision [J/OL]. Computer Science, 2015 [2018-04-13]. https://arxiv.org/abs/1502.02551

[56] Peemen M, Setio A A A, Mesman B, et al. Memory-centric accelerator design for convolutional neural networks [C] // Proc of the 31st Int Conf on Computer Design. Piscataway, NJ: IEEE, 2013: 13-19

[57] Zhang Chen, Li Peng, Sun Guangyu, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks [C] //Proc of the 23rd ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2015: 161-170

[58] Albericio J, Judd P, Hetherington T, et al. Cnvlutin: Ineffectual-neuron-free deep neural network computing [C] // Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 1-13

[59] Zhang Shijin, Du Zidong, Zhang Lei, et al. Cambricon-X: An accelerator for sparse neural networks [C] //Proc of the 49th IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2016: No. 20

[60] Han Song, Liu Xingyu, Mao Huizi, et al. EIE: Efficient inference engine on compressed deep neural network [C] // Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 243-254

[61] Yu Jiecao, Lukefahr A, Palframan D, et al. Scalpel: Customizing DNN pruning to the underlying hardware parallelism [C] //Proc of the 44th Annual Int Symp on Computer Architecture. New York: ACM, 2017: 548-560

Chen Guilin, born in 1994. MSc. Student member of CCF. His main research interests include architecture, machine learning and neural network accelerator.

Ma Sheng, born in 1986. PhD, assistant professor. Member of CCF. His main research interests include computer architecture, on-chip networks and arithmetic unit designs.

Guo Yang, born in 1971. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include low power VLSI circuits, microprocessor design and verification, and electronic design automation (EDA) techniques for VLSI circuits. (guoyang@nudt.edu.cn)