Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Artificial Intelligence—Review

深度神经网络加速器体系结构概述

陈怡然[a,*]，谢源[b]，宋凌皓[a]，陈凡[a]，唐天琪[b]

[a] Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA
[b] Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560, USA

ARTICLE INFO

Article history:
Received 1 June 2019
Revised 6 September 2019
Accepted 9 January 2020
Available online 29 January 2020

关键词
深度神经网络
特定领域体系结构
加速器

摘要

最近，由于可使用的大数据和计算能力的快速增长，人工智能重新获得了巨大的关注和投资。机器学习（ML）方法已成功应用于解决学术界和工业界的许多问题。尽管大数据应用的高速增长为ML的发展提供动力，但它也给传统计算机系统带来了数据处理速度和可扩展性方面的严峻挑战。专门为AI应用程序设计的计算平台已经从对冯·诺依曼（von Neumann）平台的补充发展到必备的独立技术解决方案。这些平台属于更大的类别，被称为“专有域计算”，专注于针对AI的特定定制。在本文中，我们特别总结了用于深度神经网络（DNN）的加速器设计（即DNN加速器）的最新进展。我们从计算单元、数据流优化、网络模型、基于新兴技术的体系结构以及针对新兴应用的加速器等方面讨论支持DNN执行的各种体系结构。我们还提供了有关AI芯片设计未来趋势的展望。
© 2020 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. 引言

经典哲学将人类思维过程描述为对符号的机械操纵。长期以来，人类一直试图创造出具有意识智能的人造物件，这是人工智能（AI）的最初萌芽。1950年，艾伦·图灵（Alan Turing）在数学上讨论了实现智能机器的可能性，并提出了“模仿游戏”（imitation game），后来被称为“图灵测试”（Turing test）[1]。达特茅斯夏季人工智能研究计划[2]于1956年进行，通常被认为是AI作为一个新的研究领域的正式开创性活动。在随后的几十年中，AI经历了几次起伏。最近，由于可供使用的大数据和计算能力的快速增长，人工智能重新获得了巨大的关注和投资。机器学习（ML）方法已成功应用于解决学术界[3,4]和工业界[5]中的许多问题。

机器学习算法（包括生物学上合理的模型）最初是为了明确地模拟生物学上的大脑行为[6]。人脑目前被认为是最智能的“机器”，具有极高的结构复杂性和运行效率。类似于生物神经系统，机器学习算法中的两个基本功能单元也是突触和神经元，它们分别负责信息处理和特征提取。与突触相比，还有更多类型的神经元模型，如McCulloch-Pitts [6]、Sigmoid、ReLU和Integrate-and-Fires [7]。这些神经元模型都具有某些非线性特征，这对于特征提取和神经网络训练都是必需的。后来，“生物学启发”的模型被发明为实现高级功能的数学方法[8]。一般来说，现代机器学习算法可分为两类：人工神经网络（ANN），其中数据表示为数值[9]；以及

* Corresponding author.
E-mail address: yiran.chen@duke.edu (Y. Chen)

2095-8099/© 2020 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
英文原文: Engineering 2020, 6(3): 264–274
引用本文: Yiran Chen, Yuan Xie, Linghao Song, Fan Chen, Tianqi Tang. A Survey of Accelerator Architectures for Deep Neural Networks. *Engineering*, https://doi.org/10.1016/j.eng.2020.01.007

脉冲神经网络（SNN），其中数据由脉冲表示[10]。

尽管大数据应用的高速增长为ML的发展提供动力，但它也给传统计算机系统带来了数据处理速度和可扩展性方面的严峻挑战。具体而言，传统的冯·诺依曼计算机具有单独的处理和存储部件。处理器与片外存储器之间频繁的数据移动限制了系统性能和能效，而AI应用程序中数据量的飙升进一步加剧了这种情况。专为AI应用程序设计的计算平台已经从对冯·诺依曼平台的补充发展到必备的独立技术解决方案。这些平台属于更大的类别，被称为“专有域计算”，专注于针对AI的特定定制。通过克服众所周知的“内存墙”（memory wall）[11]和“电源墙”（power wall）[12]的挑战，已经实现了数量级能效和性能的提高。最近的特定于AI的计算系统（即AI加速器）通常由大量高度并行的计算和存储单元构成。这些单元以二维方式组织，以支持神经网络（NN）中常见的矩阵向量乘法。片上网络（NoC）[13]、高带宽存储器（HBM）[14]和数据重用[15]等被用于进一步优化这些加速器中的数据流。生物逻辑理论基础、硬件设计和算法（软件）这三个层次的创新是AI加速器的三个基石。本文将总结AI加速器在数据中心[5,16,17]和边缘设备[18–20]上的最新进展。

除了传统的CMOS设计之外，最近在AI加速器设计中还探索了新兴的非易失性存储器的应用，如金属氧化物阻性随机存取存储器（ReRAM）等。这些新兴的存储器具有高存储密度和快速访问的特点，并且具有实现内存计算的潜力[21–23]。具体而言，ReRAM阵列不仅可以存储神经网络，而且还能够以模拟方式执行原位（*in situ*）矩阵矢量乘法。与最先进的CMOS设计相比，基于ReRAM的AI加速器由于模拟计算的低功耗特性，可以实现3~4个数量级的更高计算效率[24]。另一方面，由于机器学习算法对噪声和错误表现出极大的抵抗力，模拟运算的噪声在很大程度上可以被机器学习算法所容忍。然而，ReRAM交叉阵列中的模拟信号与加速器中其他数字单元中的数字值之间的转换需要数模转换器（DAC）和模数转换器（ADC），对于基于ReRAM的NN加速器，这要花费高达66.4%的功耗和73.2%的面积[25]。

在本文中，我们主要关注人工神经网络。我们特别总结了用于深度神经网络（DNN）的加速器设计的最新进展。我们从计算单元、数据流优化、网络模型等方面讨论支持DNN执行的各种体系结构。本文的组织如下：第2节介绍了机器学习和深度神经网络的基础；第3节和第4节分别介绍了几种代表性的DNN片上加速器和独立加速器；第5节描述了各种基于新兴内存技术的DNN加速器；第6节简要总结了新兴应用程序的DNN加速器；第7节提供了我们对AI芯片设计的未来趋势的展望。

2. 背景

在本节中，我们将介绍有关DNN的一些背景以及阅读本文所需的一些重要概念。我们还将简要介绍新兴的ReRAM及其在神经计算中的应用。

2.1. 深度神经网络的推理和训练

通常来说，DNN是参数化函数，需要高维输入以做出一些有用的预测，如分类。这种预测过程称为推理。为了获得有意义的参数集，我们需要在训练数据集上进行DNN的训练，并通过诸如随机梯度下降（SGD）之类的方法对参数进行优化，以最大限度地减少某些预定义的损失函数。在每个训练步骤中，首先执行前向传播以计算损耗，然后进行反向传播以反向传播错误，最后，计算并累积每个参数的梯度。为了完全优化大规模DNN，训练过程可能需要上百万步或更多。

DNN通常是神经网络层的堆叠。如果我们将第l_{th}层表示为函数f_l，则这个L层DNN的推理可以表示为：

$$f(x)=f_{l-1}\circ f_{l-2}\circ\ldots\circ f_2\circ f_1(x) \qquad (1)$$

式中，x是输入。在这种情况下，每一层的输出仅供下一层使用，并且整个计算没有回溯。DNN推理的数据流采用链的形式，可以在硬件中有效地加速，而无需额外的内存需求。前馈神经网络和递归神经网络（RNN）均适用此属性。“循环”结构可以看作是可变长度的前

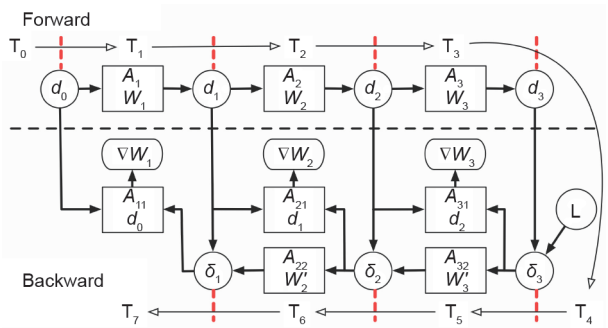

图1. Pipelayer中的DNN训练数据流[22]。每个箭头代表一个数据依赖性。

馈结构，可以暂时复用某一层权重，并且数据流仍然形成一条链。

在DNN训练中，数据依赖性的深度是推理深度的两倍。尽管前向传播的数据流与推理相同，但是反向转播以相反的顺序执行层计算。此外，前向传播中各层的输出在反向传播中被重新使用以计算误差（由于反向传播的链法则），从而导致许多长的数据依赖性。图1说明了训练数据流与推论有何不同。DNN可能包括卷积层、全连接层（带批处理的矩阵乘法）、一些逐点操作层，如ReLU、Sigmoid、最大池化和批归一化。反向传播可能具有形式不同于前向传播的逐点操作。矩阵乘法和卷积在反向传播中也保持不变。主要区别在于它们分别在转置权重矩阵和旋转卷积核上执行。

2.2. 计算模式

尽管DNN可能包含许多类型的层，但是矩阵乘法和卷积占了90%以上的运算，并且是DNN加速器设计的主要优化目标。对于矩阵乘法，如果我们分别使用I_c, O_c, B表示输入通道数、输出通道数和批处理大小，则计算可写为：

$$\text{output}_{b,o_c} = \sum_{i_c=0}^{I_c} \text{input}_{b,i_c} \times \text{weight}_{i_c,o_c} \quad (2)$$

式中，i_c是输入通道的索引；o_c是输出通道的索引；b是批处理样本的索引。满足$0 \le b < B$, $0 \le o_c < O_c$。矩阵乘法中涉及的数据复用是每个输入都用于所有输出通道，每个权重都用于所有输入批次。

DNN中的卷积可以看作是矩阵乘法的扩展版本，它增加了局部连接性和平移不变性的属性。与矩阵乘法相比，在卷积中，每个输入元素被二维特征图替换，每个权重元素被二维卷积核（或滤波器）替换。然后，基于滑动窗口进行计算：如图2所示，从输入特征图的左上角开始，过滤器向右端滑动。当它到达特征图的右端时，它将移回到左端并移至下一行。正式表示如下所示：

$$\text{output}_{b,o_c,x,y} = \sum_{i_c=0}^{I_c} \sum_{i=0}^{F_h} \sum_{j=0}^{F_w} \text{input}_{b,i_c,x+i,y+j} \times \text{filter}_{o_c,i_c,i,j_c} \quad (3)$$

式中，F_h是滤波器的高度；F_w是滤波器的宽度；i是二维滤波器中行的索引；j是二维滤波器中列的索引；x是二维特征图中行的索引；y是特征二维图中列的索引。满足$0 \le b < B$, $0 \le o_c < O_c$, $0 \le x < O_h$, $0 \le y < O_w$（O_h是输出特征图的高度；O_w是输出特征图的宽度）。

为了提供平移不变性，将相同的卷积滤波器重复应用于输入特征图的所有部分，从而使卷积中的数据复用模式比矩阵乘法复杂得多。为了简化硬件实现，最好以两级结构查看二维滑动窗口：第一层是向下滑动的行窗口，以提供行间数据复用；第二层是向右滑动的元素窗口，以提供行内数据复用。

尽管矩阵乘法和卷积的计算模式非常不同，但它们实际上可以相互转换。因此，为一种计算类型设计的加速器仍然可以支持另一种，尽管效率可能不高。如图3所示，可以通过Toeplitz矩阵将卷积转换为矩阵乘法，而代价是引入冗余数据。另一方面，矩阵乘法可以视为一种卷积形式，其中，$O_h = O_w = F_w = F_h = 1$。特征图和滤波器都被简化为单个元素。

2.3. 阻性存储器

忆阻器（Memristor），又名阻性随机存储器（ReRAM），是一种新兴的非易失性存储器，它使用单元电阻来存储信息。2008年，惠普实验室报道了他们基于TiO_2薄膜器件的纳米级忆阻器的发现[27]。从那时起，许多阻性材料和结构被发现或重新发现。

如图4（a）所示，每个ReRAM单元都有一个夹在顶部电极（TE）和底部电极（BE）之间的金属氧化物层。忆阻器的电阻可以通过施加具有适当脉冲宽度或幅度的电流或电压来编程。特别地，存储在单元中的数据可以相应地由电阻状态表示：低电阻状态（LRS）表示位“1”，高电阻状态（HRS）表示位“0”。在读取操作中，在

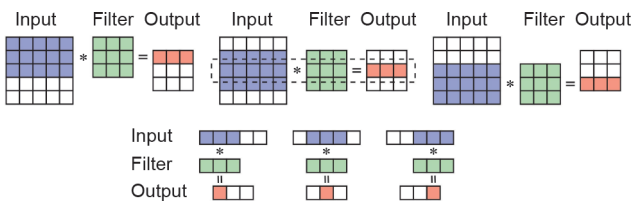

图2. 二维卷积中的两个级别的滑动窗口[26]。

器件上施加一个小的检测电压，然后由电阻确定电流的幅度。

2012年，惠普实验室提出了一种ReRAM交叉结构（crossbar），该结构展示了吸引人的能力，可以有效地加速神经网络中的矩阵矢量乘法。如图4（b）所示，矢量由字线（WL）上的输入信号表示。矩阵的每个元素都被编程为交叉阵列中一个单元的电导。因此，将每个位线（BL）末端的电流求和视为矩阵矢量乘法的结果。对于不能容纳在单个阵列中的大型矩阵，应将输入和输出分割并分组为多个阵列。每个整列的输出是部分和，将其水平收集并垂直求和以生成实际结果。

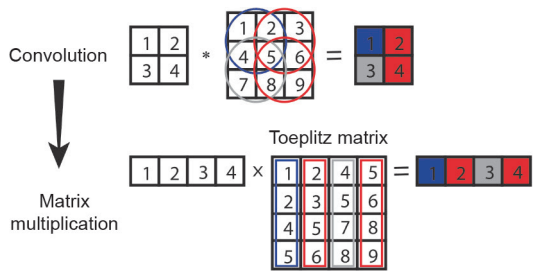

图3. 将卷积转换为Toeplitz矩阵乘法。

3. 片上加速器

在DNN加速器设计的早期阶段，加速器被设计用于加速通用处理中的近似程序[28]或用于小型神经网络[13]。尽管片上加速器的功能和性能非常有限，但它们揭示了AI专用芯片的基本思想。由于通用处理芯片的局限性，我们需要设计用于AI/DNN应用的专用芯片。

3.1. 神经处理单元

神经处理单元（neural processing unit, NPU）[28]被设计使用片上神经网络硬件来加速程序的一部分，代替在CPU上运行。

NPU的硬件设计非常简单。NPU由8个处理引擎（PE）组成，如图5所示。每个PE都执行神经元的计算，

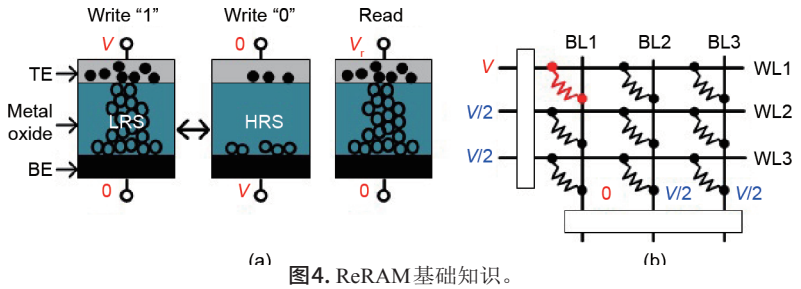

图4. ReRAM基础知识。

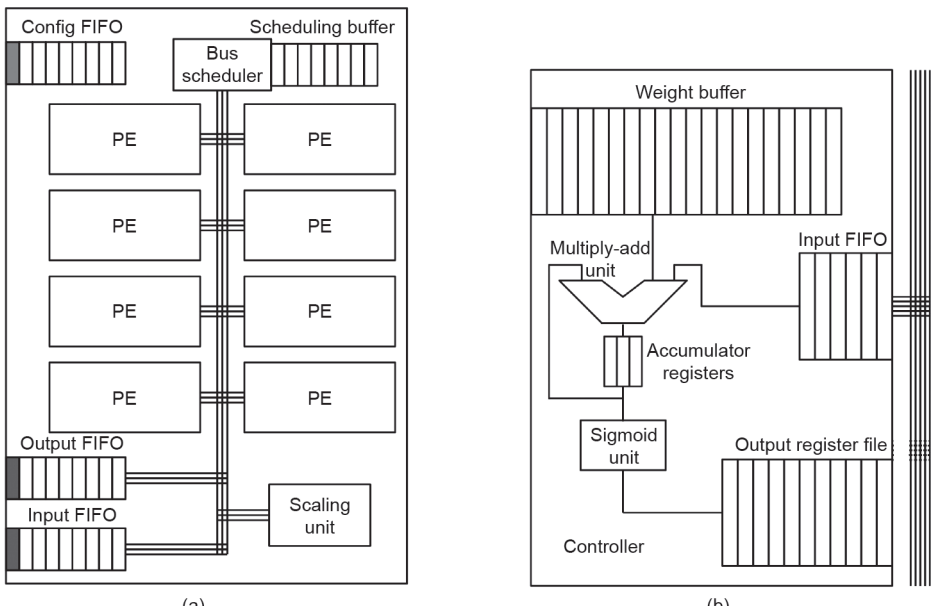

图5. 神经处理单元[28]。

即乘法、累加和sigmoid激活。因此，NPU执行的是多层感知器（MLP）神经网络的计算。

使用硬件化的MLP（即NPU）来加速某些程序段的想法非常有启发性。如果程序段符合：①经常执行，②近似执行，并且③输入和输出定义明确，则可以用NPU加速。为了在NPU上执行程序，程序员需要手动标注满足以上三个条件的程序段。然后，编译器会将程序段编译为NPU指令，并且在运行时将计算任务从CPU移到NPU。Sobel 边缘检测和FFT是此类程序段的两个示例。NPU最多可减少97%的动态CPU指令，并实现高达11.1倍的加速。

3.2. RENO——一种可重新配置的 NoC 加速器

与NPU用于通用程序加速不同，RENO [13]是用于神经网络的加速器。RENO在处理引擎设计时采用了与NPU类似的想法，如图6所示。但是，RENO的PE基于ReRAM：RENO使用ReRAM交叉结构作为基本计算单元来执行矢量矩阵乘法。每个PE包含四个ReRAM交叉结构，分别对应于正输入和负输入以及正权重和负权重的处理。在RENO中，路由器（router）用以协调PE之间的数据传输。与常规CMOS路由器不同，RENO的路由器将模拟（analog）中间计算结果从前一个神经元传递到后一个神经元。在RENO中，只有输入和最终输出是数字的。中间结果都是模拟的，并由模拟路由器协调。仅当在RENO和CPU之间传输数据时，才需要数据转换器（模数转换器ADC和数模转换器DAC）。

RENO支持多层感知器（MLP）和自动关联存储器（AAM）的处理，并且相应的指令专为RENO和CPU的流水线设计。由于RENO是片上设计，因此支持的应用程序有限制。RENO支持处理小型数据集，即UCI机器学习存储库[29]和定制的MNIST [30]。

4. 独立的 DNN/CNN 加速器

对于广泛使用的DNN和CNN（卷积神经网络）应用，独立的专有域的加速器在云和边缘场景中均取得了巨大的成功。与通用CPU和GPU相比，这些定制架构可提供更好的性能和更高的能效。定制体系结构通常需要对目标应用有深刻的了解。在设计中仔细分析并利用数据流（或数据复用模式），以减少片外存储器访问并提高系统效率。

在本节中，我们将分别以电脑（DianNao）系列[31]和张量处理单元（TPU）[5]作为学术界和工业实例来说明独立加速器的设计并讨论数据流分析。

4.1. 电脑系列——一个学术界的典型

电脑（DianNao）系列包括表1中列出的多个加速器。DianNao是该系列的第一个设计，它由以下组件组成，如图7所示：

（1）执行计算的神经功能单元（NFU）；

（2）输入神经元的输入缓冲区（NB_{in}）；

（3）输出神经元的输出缓冲器（NB_{out}）；

（4）用于突触权重（SB）的突触缓冲；以及

（5）控制逻辑（CP）。

其中，包括乘法器、加法器树和非线性功能单元的NFU被设计为流水线。暂存器（scratchpad memory）有

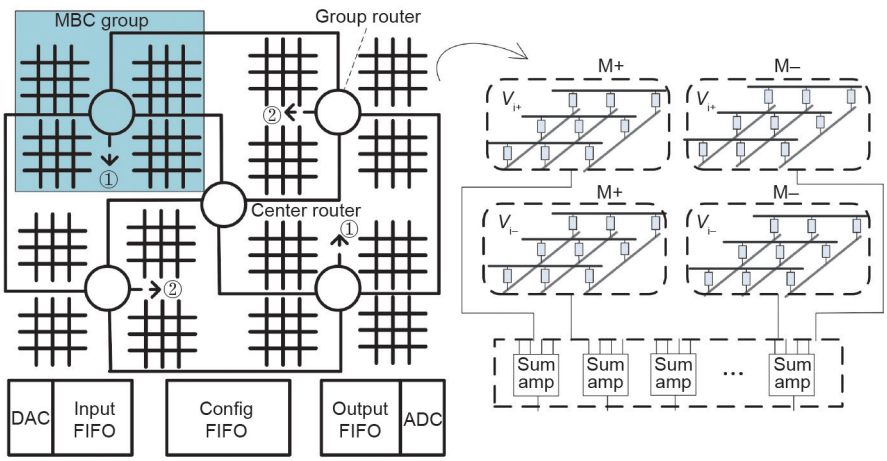

图6. RENO 架构[13]。

别于常规的高速缓存（cache），这里用作片上存储，因为它可以由编译器控制并且易于利用数据局部性。

尽管高效的计算单元对于DNN加速器很重要，但低效的内存传输也会影响系统的吞吐量和能效。电脑系列引入了特殊设计，以最大限度地减少内存传输延迟并提高系统效率。DaDianNao [16]针对数据中心场景，集成了大型片上eDRAM，以避免较长的主存储器访问时间。同样的原则也适用于嵌入式方案。ShiDianNao [19]是专用于CNN应用程序的DNN加速器。由于权重复用，CNN的内存占用量比其他DNN小得多。当CNN模型较小时，可以将所有CNN参数映射到较小的片上SRAM。通过这种方式，ShiDianNao避免了昂贵的片外DRAM访问，并且与DianNao相比实现了60倍的能效。

PuDianNao [17]设计用于多种机器学习应用。除了DNN，它还支持其他代表性的机器学习算法，如*k*-平均（*k*-means）和分类树（classification tree）。为了处理这些工作负荷的不同数据访问模式，PudianNao在其体系结构中引入了具有不同复用距离的数据的冷缓冲区和热缓冲区。此外，还引入了包括循环展开（loop unrolling）、循环平铺（loop tiling）和缓存区块化（cache blocking）在内的编译技术，作为软硬件协同设计方法，以提高片上数据的复用率和PE利用率。

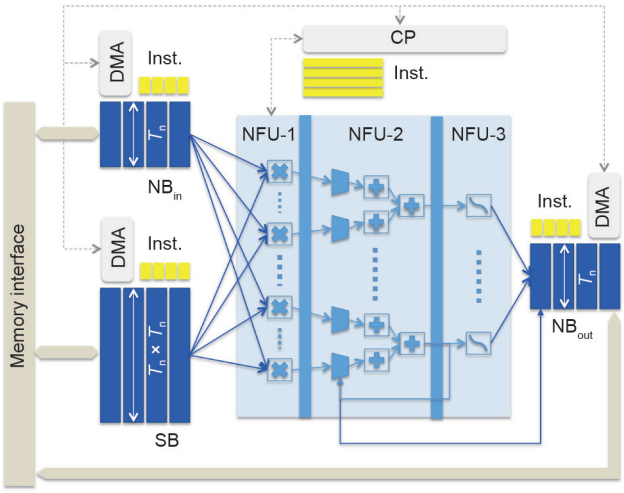

图7. 电脑架构[18]。

除了独立的加速器之外，电脑系列还提出了一种称为Cambricon [32]的领域特定指令集架构（ISA），以支持广泛的神经网络应用。Cambricon是一种存-储（load-store）体系结构，集成了标量、向量、矩阵、逻辑、数据传输和控制指令。ISA设计考虑了数据并行性、定制化矢量/矩阵指令以及暂存器的使用。

Cambricon系列的后续产品引入了支持稀疏神经网络的方法。其他加速器支持更复杂的NN工作负荷，如LSTM和GAN。这些工作将在第6节中详细讨论。

4.2. TPU——一个工业界的典型

Google在2017年发布的第一篇TPU论文（TPU1）[5]，如图8所示，特别注意其使用的脉动阵列（Systolic Array）。TPU1专注于推理任务，自2015年以来就已在Google的数据中心中进行了部署。脉动阵列的结构可视为专用的权重固定数据流或二维但指令多数据（2D SIMD）架构。之后，在Google I/O'17 [33]中，Google宣布了其云TPU（也称为TPU2），它可以处理数据中心中的训练和推理。TPU2也采用脉动阵列，并引入了向量处理单元。在Google I/O'18 [34]中，Google宣布了TPU3，其具备液体冷却功能这一特点。在Google NEXT'18 [35]中，Google宣布了其边缘TPU，其目标是物联网（IoT）的推理任务。

4.3. 数据流分析和架构设计

通常来说，DNN/CNN需要大量内存空间。对于大型和复杂的DNN/CNN模型，不太可能将整个模型映射到芯片上。由于有限的片外带宽，提高片上数据复用率和减少片外数据传输对于提高计算效率至关重要。在体系结构设计中，需要分析并且特别考虑执行数据流（dataflow）。如图9所示，Eyeriss [15,36]探索了不同的神经网络数据流，包括输入固定（IS）、输出固定（OS）、权重固定（WS）和无本地重用（NLR）在空间架构，然后提出了行固定（RS）数据流，以增强数据复用。

表1 电脑系列加速器[31]

Name	Process (nm)	Peak performance (GOP·s^{-1})	Peak power (W)	Area (mm^2)	Applications
DianNao	65	452	0.485	3.02	DNN
DaDianNao	28	5585	15.970	67.70	DNN
ShiDianNao	65	194	0.320	4.86	CNN
PuDianNao	65	1056	0.596	3.51	7 ML algorithms

GOP: group of picture.

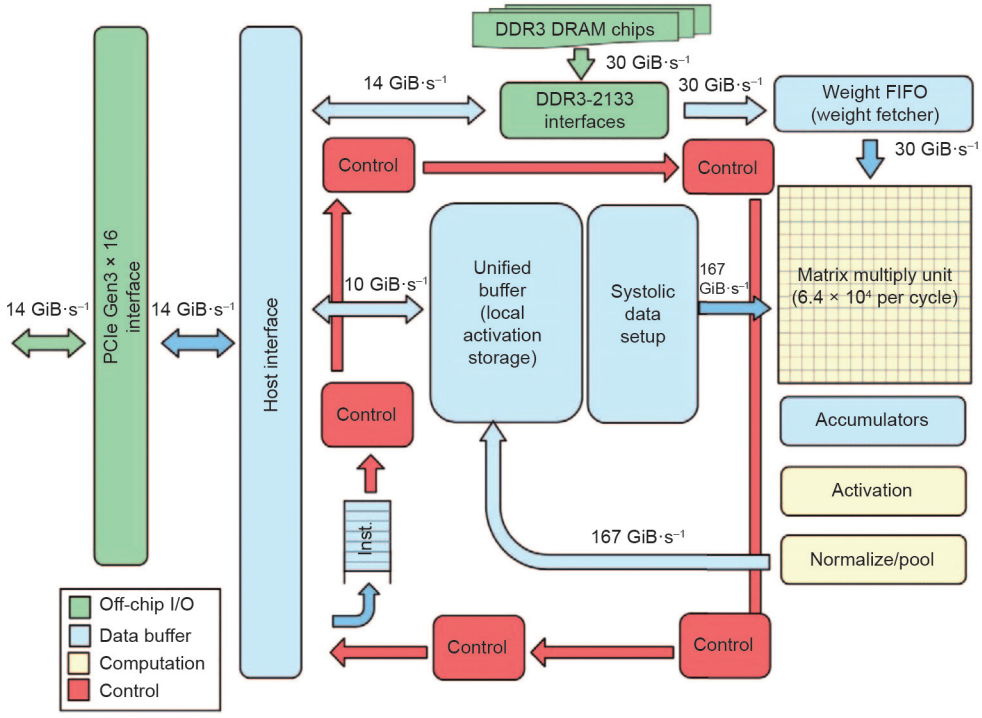

图8. TPU框图[5]。

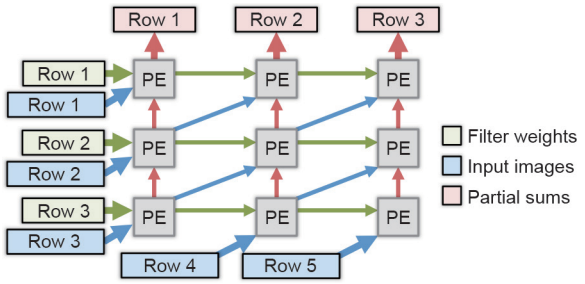

图9. 行固定（row stationary）数据流[15,36]。

高效的数据流设计也启发了AI芯片行业的许多实用设计。例如，WaveComputing具有基于粗粒度可重配置阵列（CGRA）的数据流处理器[37]。又如，GraphCore专注于图体系结构[38]，并声称在AI工作负荷方面能够实现比传统标量处理器和矢量处理器更高的性能。

5. 基于新兴存储器的加速器

ReRAM [27]和混合存储立方体（HMC）[39]是具有代表性的新兴存储技术和可实现内存中处理（PIM）的存储结构。CPU和片外存储器之间的数据移动比浮点操作消耗的能量大两个数量级[40]。PIM可以极大地减少计算中的数据移动。DNN加速器可以从ReRAM和HMC中获得这些好处，并应用PIM来加速DNN执行。

5.1. 基于 ReRAM 的 DNN 加速器

利用ReRAM进行DNN加速的关键思想是将ReRAM阵列用作矢量矩阵乘法的计算引擎[41,42]，如第2.3节所述。PRIME [21]、ISAAC [25]、PipeLayer [22]是三个基于ReRAM的代表性DNN加速器。

PRIME [21]的架构如图10所示。PRIME修改了主存储器（ReRAM）设计，以进行数据存储和计算。在PRIME中，字线（WL）解码器和驱动器配置有多级电压源，因此输入特征图可以在计算中被输入存储器阵列进行计算。列多路复用器（column multiplexer）配置有模拟减法和Sigmoid电路，因此将两个阵列的部分结果合并并发送到非线性激活单元（sigmoid）。感测放大器（sense amplifier）还可以重新配置感测分辨率，并执行模数转换器的功能。

ISAAC [25]提出了一种用于ReRAM中的NN处理的块内流水线（intra-tile pipeline）设计，如图11所示。其流水线设计结合了数据编码和计算。IMA是基于ReRAM的原位（*in situ*）乘法累加单元。在流水线中，在第一个周期，数据从eDRAM读取到计算区块中。ISAAC中的数据格式是16位定点。在计算中，在每个周期中，将一位输入IMA，并将来自IMA的计算结果转换为数字格式，移位1位并累加。因此，还要花费16个周期来处理输入。然后将结果应用于非线性激活，并将

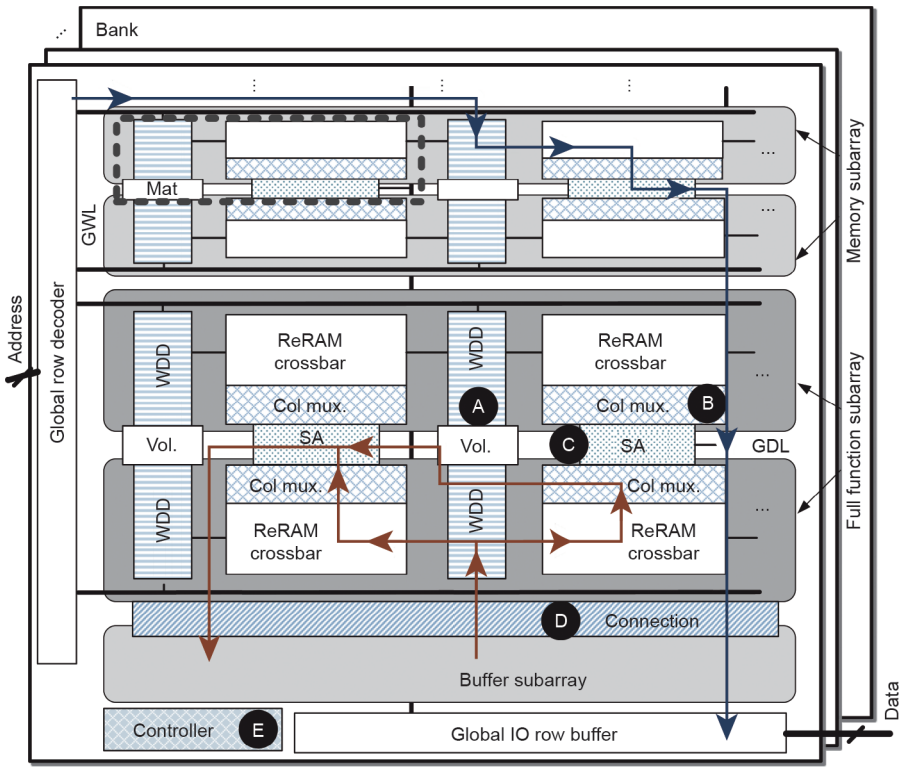

图10. PRIME 架构[21]。

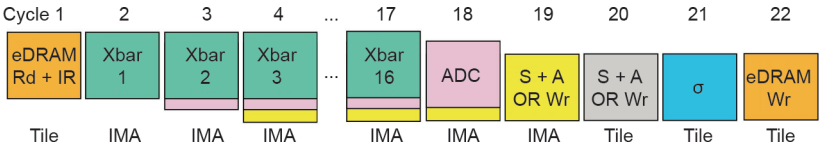

图11. ISAAC中的块内流水线 [25]。

结果写回到eDRAM。

分块计算（tiled computation）架构是一种自然且广泛使用的处理NN的方法。有必要探索一些粗粒度的设计以提高加速器的吞吐量。PipeLayer [22]为分块计算体系结构引入了层内并行（intra-layer parallelism）和层间流水线（inter-layer pipeline），以提高吞吐量，如图11所示。对于层内并行，PipeLayer使用数据并行方案，该方案复制具有相同结构的处理单元，用相同权重以并行处理多个数据。对于层间流水线，缓冲区被用于在层之间共享数据，由此，多个层的计算可以并行处理。这种层间流水线属于模型并行方案。

5.2. 基于 HMC 的 DNN 加速器

HMC垂直集成了DRAM片层和逻辑片层。HMC提供的高存储容量、高存储带宽和低延迟可实现近数据处理（near-data processing）。在基于HMC的加速器设计中，将计算和逻辑单元放置在逻辑片层上，并将DRAM片层用于数据存储。Neurocube [43]和Tetris [44]是基于HMC的两个代表性DNN加速器。

Neurocube [43]中的整个加速器具有一个HMC和16个vaults。如图12所示，每个vault都可以视为一个子系统，该子系统由执行乘法累加（MAC）的处理引擎（PE）和用于在逻辑芯片和DRAM芯片之间进行包裹传输的路由器组成。每个vault都可以通过路由器将数据发送到目标vault，从而可以实现乱序的（out-of-order）数据到达。对于每个PE，如果缓冲区（16个条目）已填充满数据，则将进行计算。

Tetris [44]也是在一个HMC中部署了16个PE，但是它使用空间网格（spatial mesh）来连接PE。Tetris提出了一种绕过顺序（bypassing ordering）方案，与文献[15,36]中讨论的数据固定方案相似，以提高数据的复用。为了最大限度地减少数据远程访问，Tetris也探索

了输入和输出特征图的分块。

6. 新兴应用加速器

通过应用高效的NN结构，也可以提高DNN加速器的效率。例如，NN剪枝（pruning）使得模型变得小而稀疏，从而减少了片外存储器访问。NN量化（quantization）使模型可以在低精度模式下运行，从而减少了所需的存储容量和计算成本。诸如生成对抗网络（GAN）和递归神经网络（RNN）等的新兴应用对专用加速器设计提出了特殊要求。本节将讨论稀疏神经网络（6.1节）、低精度神经网络（6.2节）、生成对抗网络（6.3节）和递归神经网络（6.4节）的加速器设计。

6.1. 稀疏神经网络

先前的工作DSD [46]表明，大部分的NN连接可以被修剪为零，而不会造成精度损失或仅有很小的损失。许多相应的支持稀疏化计算架构也被提出。例如，EIE [47]、Cnvlutin [48]分别针对通过稀疏权重矩阵和稀疏特征图来加速NN的计算。但是，这些设计中采用的特殊数据格式和额外的编码/解码会带来额外的硬件开销。一些算法工作讨论了如何以硬件友好的方式（如块稀疏性）设计NN模型[45]，如图13所示。此外，一些可以处理稀疏NN中不规则内存访问和不平衡工作量的技术也被提出。例如，Cambricon-X [49]和Cambricon-S [50]通过软件/硬件的协作方法解决了稀疏NNN中的内存访问不规范问题。ReCom [51]提出了一种基于结构化权重/激活压缩的基于ReRAM的稀疏NN加速器。

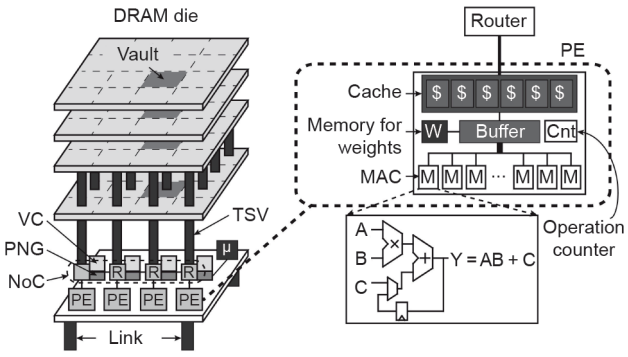

图12. Neurocube [43]架构和PE设计。

6.2. 低精度神经网络

降低数据精度或量化是提高DNN加速器计算效率的另一种可行方法。TensorRT [52]的最新结果表明，包括AlexNet、VGG、ResNet等在内的广泛使用的NN模型可以量化为8位，而不会导致推理精度损失。但是，当采用更低的精度时，这种统一的量化策略很难保持网络的精度。许多复杂的量化方案被提出，但是，这大大增加了加速器设计中量化编码/解码和工作负荷调度的硬件开销。正如我们将在以下内容中展示的那样，在进行各种优化后，数据精度与整体系统效率之间存在一个“甜点”（“sweet point”）。

（1）将权重和特征图量化为不同的精度，以实现较低的推理精度损失。这可能会更改原始数据流，并影响加速器架构，尤其是暂存器内存。

（2）不同的层或不同的数据可能采用不同的量化策略。通常，NN的第一层和最后一层需要更高的精度。这个事实增加了量化编码/解码和工作量调度的设计复杂度。

（3）通过观察数据分布特征提出新的量化方案。例如，离群值感知（outlier-aware）加速器[53]对大多数数据（权重和激活）执行密集（dense）和低精度的计算，同时有效地处理少量的稀疏和高精度离群值。

（4）新的数据格式被提出，以更好地表示低精度数据。例如，Compensated-DNN [54]引入了新的定点表示：带错误补偿的定点（fixed point with error compensation, FPEC）。这种表示有两个部分：①计算位，它们是

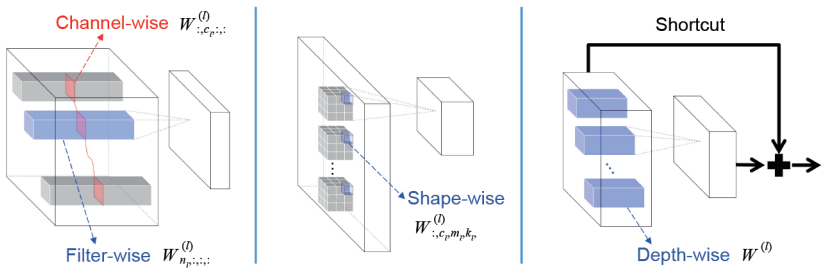

图13. 结构化稀疏性：按过滤器、形状和深度进行的稀疏[45]。

常规的定点格式；②补偿位，代表量化误差。这项工作还提出了一种低开销的稀疏补偿方案来估计MAC设计中的误差。

6.3. 生成对抗网络

与原版的DNN/CNN相比，GAN由两个NN组成，即生成器（generator）和鉴别器（discrimator）。生成器学会产生提供给鉴别器的伪造数据，而鉴别器学会辨别产生的伪造数据，目的是让生成器生成最终无法由鉴别器区分的伪造数据。这两个NN经过反复训练，并在minimax游戏中相互竞争。GAN的运算涉及一个新的运算符，称为转置卷积[也称为反卷积（deconvolution）或分数步卷积（fractionally strided convolution）]。与原始卷积相比，转置卷积执行上采样，并在特征图中插入了大量零。如果我们直接映射转置的卷积，将引入冗余计算。如果绕过零的计算，还需要一些技术来处理非结构化内存访问和不规则的数据布局。总之，与第4节中的独立DNN/CNN推理加速器相比，GAN加速器必须：①支持培训；②适应转置卷积；③优化非结构化数据访问。

ReGAN [23]提出了一种基于ReRAM的PIM GAN架构。如图14所示，专用的流水线被设计来用于逐层计算以增加系统吞吐量。为了进一步提高训练效率，空间并行和计算共享这两种技术被提出。LerGAN [55]提出了一种免零（zero-free）数据重塑方案，以消除基于ReRAM的PIM GAN架构中的零相关计算，还提出了一种可重配置的互连方案，以减少数据传输开销。

对于基于CMOS的GAN加速器，先前的工作[56]提出了针对GAN中不同步骤的有效数据流，即用于前向/反向传播的免零输出固定（zero free output stationary, ZFOST），以及用于权重更新的免零权重固定（zero free weight stationary, ZFWST）。GANAX [57]提出了一个统一的SIMD-MIMD加速器，以最大化生成器和鉴别器的效率：由于生成器中零的插入，因此选择性执行中使用SIMD-MIMD模式，而使用纯SIMD模式来操作鉴别器中的传统CNN。

6.4. 递归神经网络

RNN有很多变体，包括门循环单元（GRU）和长期短期记忆（LSTM）。与传统的DNN/CNN相比，RNN的循环特性会导致复杂的数据依赖。

ESE [58]展示了专用于稀疏LSTM的加速器。为了确保较高的硬件利用率，一种负荷平衡感知的剪枝（load-balance-aware pruning）被提出。调度器旨在将压缩模型编码并划分为多个PE，以实现并行性，并调度LSTM数据流。DNPU [59]提出了一种8.1TOPS/W可重配置的CNN-RNN片上系统（SoC）。DeltaRNN [60]利用RNN增量网络（delta network）更新方法来减少内存访问：仅当神经元的激活变化超过增量阈值时，神经元的输出才会更新。

7. DNN 加速器的未来

在本节中，我们将分享有关DNN加速器的未来的观点。我们将讨论三种可能的未来趋势：①DNN训练和加速器阵列；②基于ReRAM的PIM加速器；③边缘DNN加速器。

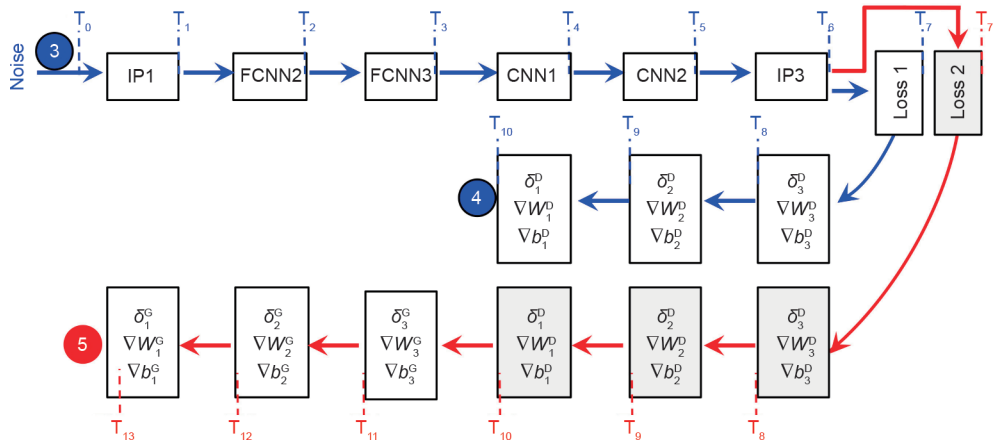

图14. ReGAN中的计算共享流水线[23]。

7.1. DNN 训练和加速器阵列

当前，几乎所有的DNN加速器架构都集中在对加速器自身内部以及DNN推理的优化上，很少有人考虑提供训练支持[22]。推理的前提是我们已经在部署DNN模型之前对其进行了训练。然而很少有支持DNN训练的加速器架构。随着训练数据集和NN的大小增加，单个加速器不再能够支持大型DNN的训练。我们不可避免地需要部署一组加速器阵列或多个加速器来训练DNN。

文献[61]提出了一种用于加速器阵列的DNN训练的混合并行结构。加速器之间的通信占据着加速器阵列上DNN训练的时间和能量消耗。文献[61]提出了一种通信模型，以识别在何处产生了数据通信以及数据通信量有多大。基于这种通信模型，对逐层并行（layer-wise parallelism）进行了优化，以最大限度地减少总通信量并提高系统性能和能效。

7.2. 基于 ReRAM 的 PIM 加速器

当前基于ReRAM的加速器（如[21,22,25,62]）假设了理想的忆阻器单元。但是，现实中的挑战，如制程差异（process variation）[63,64]、电路噪声[65,66]、保留和耐久性（retention and endurance）问题[67–69]，极大地阻碍了基于ReRAM的加速器的实现。除了文献[70]以外，基于ReRAM的加速器和高级架构（如PIM）的硅证明（silicon proof）也很少。在基于ReRAM的实际DNN加速器设计中，必须考虑这些非理想因素。

7.3. 边缘 DNN 加速器

在边缘-云（edge-cloud）DNN应用中，计算和内存密集型部分（如训练）通常被移植到云中功能强大的GPU上进行计算。在边缘设备（如IoT或移动设备）上仅部署了一些轻量推理模型。

随着数据采集规模的迅速增长，人们希望拥有一些能够针对某些任务自适应学习或微调其DNN模型的智能边缘设备。例如，在监视用户健康的可穿戴应用中，由于大量的数据通信开销和隐私问题，需要在本地调整CNN模型而不是将感测到的健康数据发送回云。在机器人、无人机和自动驾驶汽车等其他应用中，静态训练的模型无法有效地应对随时间变化的环境条件。

然而将大量环境数据发送到云端进行增量训练引起的漫长数据传输延迟通常是不可接受的。更重要的是，许多现实生活场景要求实时执行多个任务和动态适应能力[58]。然而，由于边缘设备的有限的计算资源和功率预算，在边缘设备上进行学习非常具有挑战性。RedEye[71]是用于边缘DNN处理的加速器，其中计算与传感集成在一起。为边缘DNN设计轻量级、实时且节能的体系结构是下一步的重要研究方向。

Acknowledgement

This work was supported in part by the National Science Foundations (1822085, 1725456, 1816833, 1500848, 1719160, 1725447), the Computing and Communication Foundations (1740352), the Nanoelectronics COmputing Research in the Semiconductor Research Corporation (NC-2766-A), the Center for Research on Intelligent Storage and Processing-in-memory, one of six centers in The Joint University Microelectronics Program, a SRC program sponsored by Defense Advanced Research Projects Agency.

Compliance with ethics guidelines

Yiran Chen, Yuan Xie, Linghao Song, Fan Chen, and Tianqi Tang declare that they have no conflicts of interest or financial conflicts to disclose.

References

[1] Turing AM. Computing machinery and intelligence. Mind 1950;LIX(236):433–60.
[2] McCarthy J, Minsky ML, Rochester N, Shannon CE. A proposal for the Dartmouth summer research project on artificial intelligence: August 31, 1955. AI Mag 2006;27(4):12.
[3] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM 2017;60(6):84–90.
[4] Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. 2014. arXiv:1406.1078.
[5] Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, et al. In-datacenter performance analysis of a tensor processing unit. In: Proceedings of 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture; 2017 Jun 24–28; Toronto, ON, Canada; 2017. p. 1–12.
[6] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 1943;5(4):115–33.
[7] Keener JP, Hoppensteadt FC, Rinzel J. Integrate-and-fire models of nerve membrane response to oscillatory input. SIAM J Appl Math 1981;41(3):503–17.
[8] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. Neural Comput 1989;1(4):541–51.
[9] Pino RE, Li H, Chen Y, Hu M, Liu B. Statistical memristor modeling and case study in neuromorphic computing. In: Proceedings of DAC Design Automation Conference 2012; 2012 Jun 3–7; San Francisco, CA,USA; 2012. p. 585–90.
[10] Maass W. Networks of spiking neurons: the third generation of neural network models. Neural Networks 1997;10(9):1659–71.
[11] Wulf WA, McKee SA. Hitting the memory wall: implications of the obvious. ACM SIGARCH Comput Archit News 1995;23(1):20–4.
[12] Guo X, Ipek E, Soyata T. Resistive computation: avoiding the power wall

with low-leakage, STT-MRAM based computing. In: Proceedings of the 37th Annual International Symposium on Computer Architecture; 2010 Jun 19–23; Saint-Malo, France; 2010. p. 371–82.
[13] Liu X, Mao M, Liu B, Li H, Chen Y, Li B, et al. RENO: a high-efficient reconfigurable neuromorphic computing accelerator design. In: Proceedings of 2015 52nd ACM/EDAC/IEEE Design Automation Conference; 2015 Jun 8–12; San Francisco, CA, USA; 2015. p. 1–6.
[14] Jiang L, Kim M, Wen W, Wang D. XNOR-POP: a processing-in-memory architecture for binary convolutional neural networks in Wide-IO2 DRAMs. In: Proceedings of 2017 IEEE/ACM International Symposium on Low Power Electronics and Design; 2017 Jul 24–26; Taipei, China; 2017. p. 1–6.
[15] Chen YH, Krishna T, Emer JS, Sze V. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE J Solid-State Circuits 2017;52(1):127–38.
[16] Chen Y, Luo T, Liu S, Zhang S, He L, Wang J, et al. DaDianNao: a machine-learning supercomputer. In: Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture; 2014 Dec 13–17; Cambridge, UK; 2014. P.609–22.
[17] Liu D, Chen T, Liu S, Zhou J, Zhou S, Teman O, et al. PuDianNao: a polyvalent machine learning accelerator. In: Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems; 2015 Mar 14–18; Istanbul, Turkey; 2015. p. 369–81.
[18] Chen T, Du Z, Sun N, Wang J, Wu C, Chen Y, et al. DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems; 2014 March 1–5; Salt Lake City, Utah, USA; 2014. p. 269–84.
[19] Du Z, Fasthuber R, Chen T, Ienne P, Li L, Luo T, et al. ShiDianNao: shifting vision processing closer to the sensor. In: Proceedings of the 42nd Annual International Symposium on Computer Architecture ; 2015 Jun 13–17; Portland, OR, USA; 2015. p. 92–104.
[20] Akopyan F, Sawada J, Cassidy A, Alvarez-Icaza R, Arthur J, Merolla P, et al. Truenorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. IEEE Trans Comput Aided Des Integr Circ Syst 2015;34(10):1537–57.
[21] Chi P, Li S, Xu C, Zhang T, Zhao J, Liu Y, et al. PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. SIGARCH Comput Archit News 2016;44(3):27–39.
[22] Song L, Qian X, Li H, Chen Y. Pipelayer: a pipelined ReRAM-based accelerator for deep learning. In: Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture; 2017 Feb 4–8; Austin, TX, USA; 2017.
[23] Chen F, Song L, Chen Y. ReGAN: a pipelined ReRAM-based accelerator for generative adversarial networks. In: Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference; 2018 Jan 22–25; Jeju, Korea; 2018.
[24] Liu C, Yang Q, Yan B, Yang J, Du X, Zhu W, et al. A memristor crossbar based computing engine optimized for high speed and accuracy. In: Proceedings of the 2016 IEEE Computer Society Annual Symposium on VLSI; 2016 Jul 11–13; Pittsburgh, PA, USA; 2016. p. 110–5.
[25] Shafiee A, Nag A, Muralimanohar N, Balasubramonian R, Strachan JP, Hu M, et al. ISAAC: a convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars. SIGARCH Comput Archit News 2016;44(3):14–26.
[26] Qiao X, Cao X, Yang H, Song L, Li H. Atomlayer: a universal reRAM-based CNN accelerator with atomic layer computation. In: Proceedings of the 55th Annual Design Automation Conference; 2018 Jun 24–29; San Francisco, CA, USA; 2018.
[27] Strukov DB, Snider GS, Stewart DR, Williams RS. The missing memristor found. Nature 2008;453:80–3.
[28] Esmaeilzadeh H, Sampson A, Ceze L, Burger D. Neural acceleration for general-purpose approximate programs. Commun ACM 2014;58(1):105–15.
[29] UCI machine learning repository [Internet]. Irvine: University of California; [cited 2019 Jan 18]. Available from: http://archive.ics.uci.edu/ml/
[30] LeCun Y, Cortes C, Burges CJC. The mnist database [Internet]. [cited 2019 Jan 18]. Available from: http://yann.lecun.com/exdb/mnist/
[31] Chen Y, Chen T, Xu Z, Sun N, Temam O. DianNao family: energy-efficient hardware accelerators for machine learning. Commun ACM 2016;59(11):105–12.
[32] Liu S, Du Z, Tao J, Han D, Luo T, Xie Y, et al. Cambricon: an instruction set architecture for neural networks. In: Proceedings of the 43rd International Symposium on Computer Architecture; 2016 Jun 18–22; Seoul, Korea; 2016. p. 393–405.
[33] Google I/O'17 [Internet]. California: Google [cited 2019 Jan 18]. Available from: https://events.google.com/io2017/.
[34] Google I/O'18 [Internet]. California: Google [cited 2019 Jan 18]. Available from: https://events.google.com/io2018/.
[35] Google Cloud Next'18 [Internet]. California: Google [cited 2019 Jan 18]. Available from: https://cloud.withgoogle.com/next18/sf/.
[36] Chen YH, Emer J, Sze V. Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks. In: Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture; 2016 Jun 18–22; Seoul, South Korea; 2016. p. 367–79.
[37] HC29 (2017) [Internet]. Hot chips; [cited 2019 Jan 18]. Available from: https://www.hotchips.org/archives/2010s/hc29/.
[38] GraphCore [Internet]. Bristol :GraphCore [cited 2019 Jan 18]. Available from: https://www.graphcore.ai/technology.
[39] Pawlowski JT. Hybrid memory cube (HMC). In: Proceedings of the 2011 IEEE Hot Chips 23 Symposium; 2011 Aug 17–19; Stanford, CA, USA; 2011.
[40] Farmahini-Farahani A, Ahn JH, Morrow K, Kim NS. NDA: near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules. In: Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture; 2015 Feb 7–11; Burlingame, CA, USA; 2015. p. 283–95.
[41] Hu M, Li H, Wu Q, Rose GS. Hardware realization of BSB recall function using memristor crossbar arrays. Proceedings of the 49th Annual Design Automation Conference; 2012 Jun 3–7; San Francisco, CA, USA; 2012. p. 498–503.
[42] Hu M, Strachan JP, Li Z, Grafals EM, Davila N, Graves C, et al. Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix–vector multiplication. In: Proceedings of the 53nd Annual Design Automation Conference; 2016 Jun 5–9; Austin, TX, USA; 2016. p. 1–6.
[43] Kim D, Kung J, Chai S, Yalamanchili S, Mukhopadhyay S. Neurocube: a programmable digital neuromorphic architecture with high-density 3D memory. In: Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture ; 2016 Jun 18–22; Seoul, South Korea; 2016. p. 380–92.
[44] Lu H, Wei X, Lin N, Yan G, Li X. Tetris: re-architecting convolutional neural network computation for machine learning accelerators. In: Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design; 2018 Nov 5–8; San Diego, CA, USA; 2018. p. 1–8.
[45] Wen W, Wu C, Wang Y, Chen Y, Li H. Learning structured sparsity in deep neural networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems; 2016 Dec 5–10; Barcelona, Spain; 2016. p. 2082–90.
[46] Han S, Pool J, Narang S, Mao H, Gong E, Tang S, et al. DSD: dense-sparse-dense training for deep neural networks. 2016. arXiv:1607.04381.
[47] Han S, Liu X, Mao H, Pu J, Pedram A, Horowitz MA, et al. EIE: efficient inference engine on compressed deep neural network. In: Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture; 2016 Jun 18–22; Seoul, South Korea; 2016. p. 243–54.
[48] Albericio J, Judd P, Hetherington T, Aamodt T, Jerger NE, Moshovos A. Cnvlutin: ineffectual-neuron-free deep neural network computing. In: Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture; 2016 Jun 18–22; Seoul, South Korea; 2016. p. 1–13.
[49] Zhang S, Du Z, Zhang L, Lan H, Liu S, Li L, et al. Cambricon-X: an accelerator for sparse neural networks. In: Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture; 2016 Oct 15–19; Taipei, China; 2016.
[50] Zhou X, Du Z, Guo Q, Liu S, Liu C, Wang C, et al. Cambricon-S: addressing irregularity in sparse neural networks through a cooperative software/hardware approach. In: Proceedings of the 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture; 2018 Oct 20–24; Fukuoka, Japan; 2018. p. 15–28.
[51] Ji H, Song L, Jiang L, Li HH, Chen Y. ReCom: an efficient resistive accelerator for compressed deep neural networks. In: Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition; 2018 Mar 19–23; Dresden, Germany; 2018. p. 237–40.
[52] Migacz S. 8-bit inference with TensorRT [Internet]. Available from: http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf.
[53] Park E, Kim D, Yoo S. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In: Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture; 2018 Jun 1–6; Los Angeles, CA, USA; 2018. p. 688–98.
[54] Jain S, Venkataramani S, Srinivasan V, Choi J, Chuang P, Chang L. Compensated-DNN: energy efficient low-precision deep neural networks by compensating quantization errors. In: Proceedings of the 2018 55th ACM/ESDA/IEEE Design Automation Conference; 2018 Jun 24–28; San Francisco, CA, USA; 2018. p. 1–6.
[55] Mao H, Song M, Li T, Dai Y, Shu J. LerGAN: a zero-free, low data movement and PIM-based GAN architecture. In: Proceedings of the 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture; 2018 Oct 20–24; Fukuoka, Japan; 2018. p. 669–81.
[56] Song M, Zhang J, Chen H, Li T. Towards efficient microarchitectural design for accelerating unsupervised GAN-based deep learning. In: Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture; 2018 Feb 24–28; Vienna, Austria; 2018. p. 66–77.
[57] Yazdanbakhsh A, Samadi K, Kim NS, Esmaeilzadeh H. GANAX: a unified MIMD-SIMD acceleration for generative adversarial networks. In: Proceedings of the 45th Annual International Symposium on Computer Architecture; 2018 Jun 2–6; Los Angeles, CA, USA; 2018. p. 650–61.
[58] Han S, Kang J, Mao H, Hu Y, Li X, Li Y, et al. ESE: efficient speech recognition engine with sparse LSTM on FPGA. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays; 2017 Feb 22–24; Monterey, CA, USA; 2017. p. 75–84.
[59] Shin D, Lee J, Lee J, Yoo H. 14.2 DNPU: an 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. In: Proceedings of the 2017 IEEE International Solid-State Circuits Conference; 2017 Feb 5–9; San Francisco, CA, USA; 2017. p. 240–1.
[60] Gao C, Neil D, Ceolini E, Liu SC, Delbruck T. DeltaRNN: a power-efficient

recurrent neural network accelerator. In: Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays; 2018 Feb 25–27; Monterey, CA, USA; 2018. p. 21–30.
[61] Song L, Mao J, Zhuo Y, Qian X, Li H, Chen Y. HyPar: towards hybrid parallelism for deep learning accelerator array. 2019. arXiv:1901.02067.
[62] Bojnordi MN, Ipek E. Memristive Boltzmann machine: a hardware accelerator for combinatorial optimization and deep learning. In: Proceedings of the 2016 IEEE International Symposium on High Performance Computer Architecture; 2016 Mar 12–16; Barcelona, Spain; 2016. p. 1–13.
[63] Chen A, Lin M. Variability of resistive switching memories and its impact on crossbar array performance. In: Proceedings of the 2011 International Reliability Physics Symposium; 2011 Apr 10–14; Monterey, CA, USA; 2011. p. MY.7.1–MY.7.4.
[64] Dongale TD, Patil KP, Mullani SB, More KV, Delekar SD, Patil PS, et al. Investigation of process parameter variation in the memristor based resistive random access memory (RRAM): effect of device size variations. Mater Sci Semicond Process 2015;35:174–80.
[65] Ambrogio S, Balatti S, Cubeta A, Calderoni A, Ramaswamy N, Ielmini D. Understanding switching variability and random telegraph noise in resistive RAM. In: Proceedings of the 2013 IEEE International Electron Devices Meeting; 2013 Dec 9–11; Washington, DC, USA; 2013. p. 31.5.1–4.
[66] Choi S, Yang Y, Lu W. Random telegraph noise and resistance switching analysis of oxide based resistive memory. Nanoscale 2014;6(1):400–4.
[67] Beckmann K, Holt J, Manem H, van Nostrand J, Cady NC. Nanoscale hafnium oxide RRAM devices exhibit pulse dependent behavior and multi-level resistance capability. MRS Adv 2016;1(49):3355–60.
[68] Chen YY, Goux L, Clima S, Govoreanu B, Degraeve R, Kar GS, et al. Endurance/retention trade-off on HfO_2/metalCap 1T1R bipolar RRAM. IEEE Trans Electron Dev 2013;60(3):1114–21.
[69] Wong HP, Lee H, Yu S, Chen Y, Wu Y, Chen P, et al. Metal-oxide RRAM. Proc IEEE 2012;100(6):1951–70.
[70] Xue CX, Chen WH, Liu JS, Li JF, Lin WY, Lin WE, et al. A 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN-based AI edge processors. In: Proceedings of the 2019 IEEE International Solid-State Circuits Conference; 2019 Feb 17–21; San Francisco, CA, USA; 2019. p. 388–90.
[71] LiKamWa R, Hou Y, Gao J, Polansky M, Zhong L. RedEye: analog ConvNet image sensor architecture for continuous mobile vision. In: Proceedings of the 43rd Annual International Symposium on Computer Architecture; 2016 Jun 18–22; Seoul, South Korea; 2016. p. 255–66.