合肥工业大学

2020－2021 学年 第一学期

《计算机体系结构》

深度神经网络的硬件加速

班　级　计算机科学与技术 18-3 班

学 号　2018211991

姓　名　余梓俊

成 绩　_____

2020 年 11 月 22 日

目 录

1 概述

由于大数据的可用性和计算能力的快速增长，人工智能在最近几年获得了巨大的关注和投资。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。

作为深度学习的初学者和计算机体系结构的入门者，本篇文章将粗浅地讨论深度神经网络加速器，即 DNN 加速器，设计方面的进展。文章将从计算单元、数据流优化、目标网络拓扑、新兴技术架构和新兴应用加速器等方面讨论支持 DNN 执行的各种架构，既作为一篇 DNN 加速器方面的迷你综述，也作为计算机体系结构课程的学习报告。

本文组织如下。第 2 节介绍了最大似然和最小二乘神经网络的基础。第 3 节和第 4 节分别介绍了几个有代表性的 DNN 片上和独立加速器。第 5 节包括综述总结和课程总结。

2 背景

2.1 DNN 的训练和预测

一般来说，DNN 是一个参数化的函数，它接受高维输入来进行有用的预测——也就是分类标签。这个预测过程叫做预测。为了获得一组有意义的参数，在训练数据集上执行 DNN 的训练，并且通过诸如随机梯度下降(SGD)的方法优化参数，以便最小化某个损失函数。在每个训练步骤中，首先执行向前传递以计算损失，然后执行向后传递以反向传播误差。最后，计算并累加每个参数的梯度。

要完全优化一个大规模的 DNN，培训过程可能需要一百万步或更多。DNN 通常是一堆神经网络层。如果我们将第 1 层表示为函数 f_1，那么第 1 层 DNN 的推论可由下式表示:

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

其中 x 是输入。这种情况下，每一层的输出只被下一层使用，整个计算没有回

溯。DNN 预测的数据流是以链的形式出现的，可以在硬件中有效地加速，而不需要额外的内存。这个性质对于前馈神经网络和递归神经网络都成立。“递归”结构可以被视为一个可变长度的前馈结构，具有一层权重的时间重用，数据流仍然形成一个链。

在 DNN 训练中，数据依赖性是预测的两倍。虽然前向通道的数据流与预测相同，但后向通道则以相反的顺序执行各层。此外，前向通路中每一层的输出在后向通路中被重复使用来计算误差(因为反向传播的链式规则)，导致许多长的数据依赖性。图 1 说明了训练数据流如何不同于预测。DNN 可包括卷积层、全连接层(分批矩阵乘法)和逐点运算层，如 ReLU、sigmoid、最大池和分批归一化。后向通过可能有逐点操作，其形式不同于前向通过。矩阵乘法和卷积也保持它们的计算模式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。

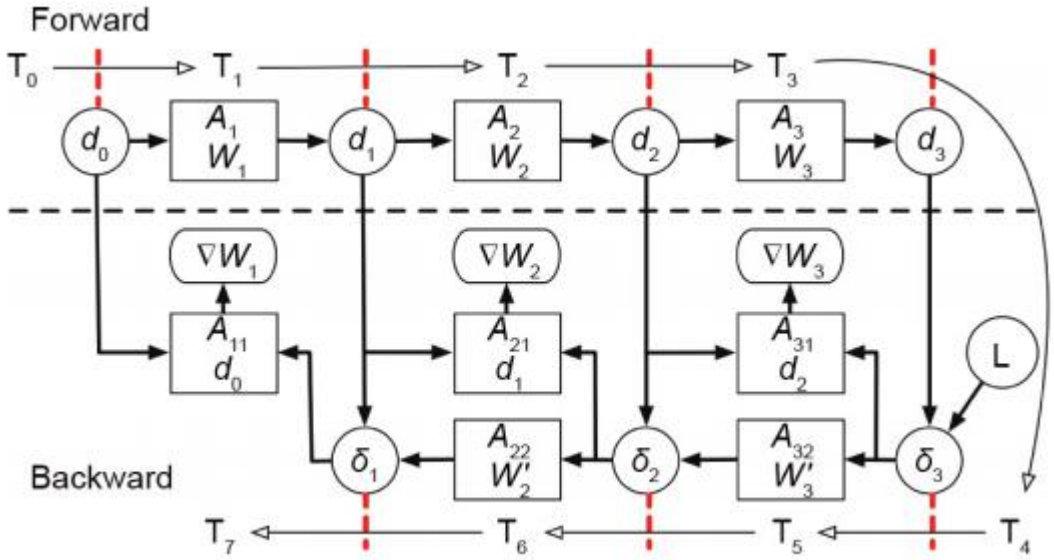

图 1 DNN 训练过程数据量

2.2 计算模式

虽然 DNN 可能包括许多类型的层，但矩阵乘法和卷积在 90%以上的运算中占主导地位，并且是 DNN 加速器设计的主要目标。对于矩阵乘法，如果我们用 Ic、Oc、B 分别表示输入通道数、输出通道数和批量，计算公式如下:

$$\text{output}_{b,o_c} = \sum_{i_c=0}^{I_c-1} \text{input}_{b,i_c} \times \text{weight}_{i_c,o_c} \qquad (2)$$

其中 ic 是输入通道的索引，Oc 是输出通道的索引，b 是一批样本的索引。对于 0b < B，0 oc < Oc。矩阵乘法涉及的数据复用是每个输入复用到所有输出通道，每个权重复用到所有输入批次。

DNN 中的卷积可以被视为矩阵乘法的扩展版本，它增加了局部连通性和平移不变性的属性。与矩阵乘法相比，在卷积中，每个输入元素由 2D 特征映射代替，每个权重元素由 2D 卷积核(或滤波器)代替。然后，计算基于滑动窗口，如图 2，从输入要素图的左上角开始，过滤器向右端滑动。当它到达要素地图的右端时，它会移回左端并移动到下一行。正式描述如下:

$$\text{output}_{b,o_c,x,y} = \sum_{i_c=0}^{I_c-1} \sum_{i=0}^{F_h-1} \sum_{j=0}^{F_w-1} \text{input}_{b,i_c,x+i,y+j} \times \text{filter}_{o_c,i_c,i,j_c} \qquad (3)$$

其中，Fh 是过滤器的高度，Fw 是过滤器的宽度，I 是 2D 过滤器中的行的索引，j 是 2D 过滤器中的列的索引，x 是 2D 要素图中的行的索引，y 是 2D 要素图中的列的索引。

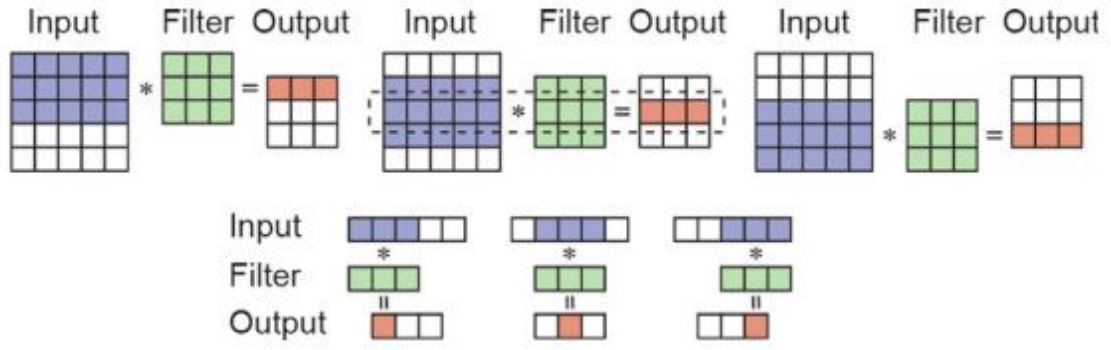

图 2 二维卷积中的两个级别的滑动窗口

为了提供平移不变性，相同的卷积滤波器被重复应用于输入特征映射的所有部分，使得卷积中的数据重用模式比矩阵乘法中的要复杂得多。为了更容易的硬件实现，最好以两级层次结构来查看 2D 滑动窗口:第一级是几行的窗口，向下滑动以提供行间数据重用，第二级是几个元素的窗口，向右滑动以提供行内数据重用。

虽然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，为一种类型的计算设计的加速器仍然可以支持另一种类型的计算，尽管这样做可能效率不高。卷积可以通过托普利兹矩阵转化为矩阵乘法，如图 3，通过引入冗余数据为代价。另一方面，矩阵乘法只是 Oh = Ow = Fw = Fh = 1 的卷积.要素图和过滤器被简化为单个元素。

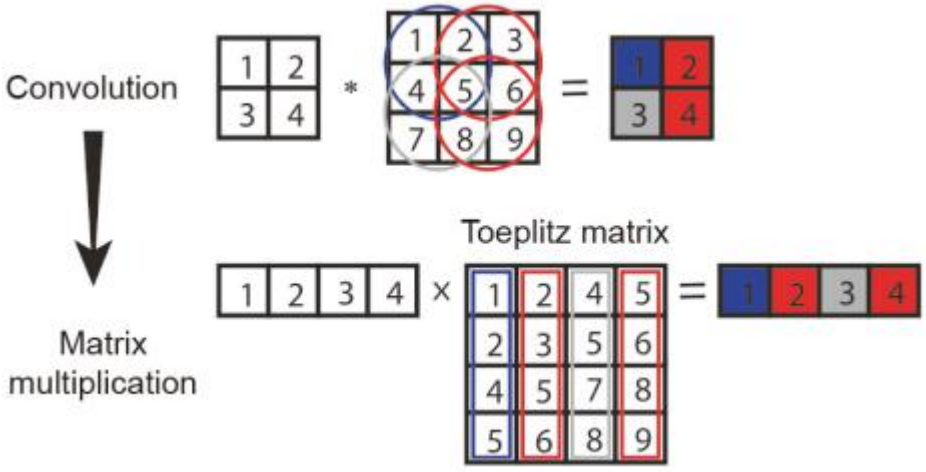

图 3 将卷积转换为 Toeplitz 矩阵乘法（ * 表示卷积）

3 片上加速器

3.1 NPU

神经处理单元（neural processing unit, NPU）[1] 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 4 所示。每个 PE 都执行神经元的计算，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。

神经处理单元采用“数据驱动并行计算”的架构，特别擅长处理视频、图像类的海量多媒体数据。以 NationalChip-GX8010 IoT AI Chip 为例，在 CPU 和 MCU 各有一个 NPU，MCU 中的 NPU 相对较小，习惯上称为 SNPU。芯片中包括了乘加、激活函数、二维数据运算、解压缩等模块。

乘加模块用于计算矩阵乘加、卷积、点乘等功能，NPU 内部有 64 个 MAC，SNPU 有 32 个。激活函数模块采用最高 12 阶参数拟合的方式实现神经网络中的激活函数，NPU 内部有 6 个 MAC，SNPU 有 3 个。二维数据运算模块用于实现对一个平面的运算，如降采样、平面数据拷贝等，NPU 内部有 1 个 MAC，SNPU 有 1 个。解压缩模块用于对权重数据的解压。为了解决物联网设备中内存带宽小的特点，在 NPU 编译器中会对神经网络中的权重进行压缩，在几乎不影响精度的情况下，可以实现 6-10 倍的压缩效果。

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。

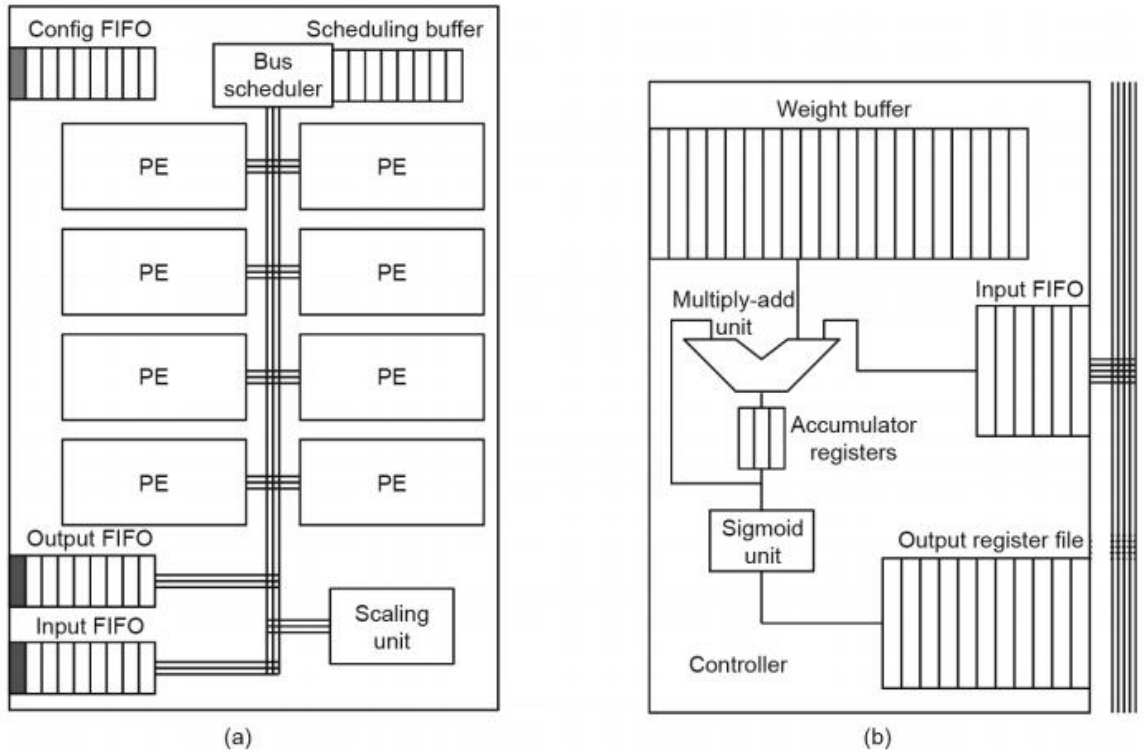

图 4 神经处理单元（NPU）架构

3.2 RENO

与 NPU 用于通用程序加速不同，RENO [2] 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 5 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。

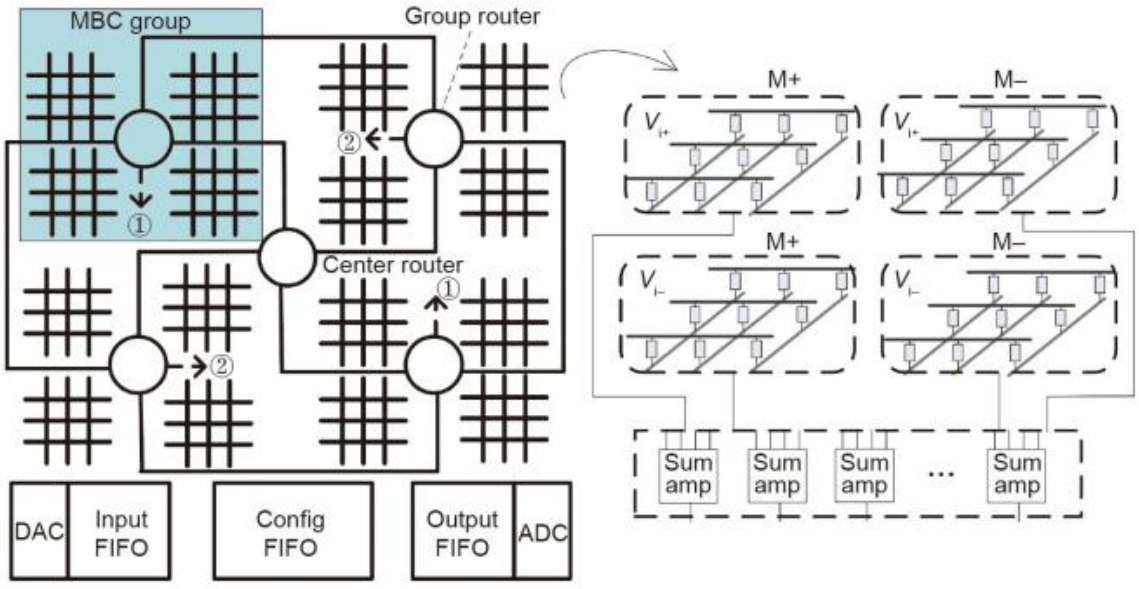

图 5 RENO 架构

4 独立加速器

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理解。

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。

4.1 DianNao 系列

中科寒武纪的前身是中国科学院计算技术研究所下的一个课题组，由陈云霁、陈天石教授领导。该课题组早在2008年就开始研究神经网络算法和芯片，并于2012年开始陆续发表研究成果。从2014年开始，寒武纪开始发表“电脑”系列论文以及“寒武纪”指令集架构：

（1）DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning

（2）DaDianNao: A Machine-Learning Supercomputer

（3）PuDianNao: A Polyvalent Machine Learning Accelerator

（4）ShiDianNao: Shifting Vision Processing Closer to the Sensor

（5）Cambricon: An Instruction Set Architecture for Neural Networks

（6）Cambricon-X: An Accelerator for Sparse Neural Networks

.....

电脑（DianNao）系列包括表1中列出的多个加速器。DianNao [3] 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（NBin）；

（3）输出神经元的输出缓冲器（NBout）；

（4）用于突触权重（SB）的突触缓冲；以及

（5）控制逻辑（CP）。

表1 电脑系列加速器

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.

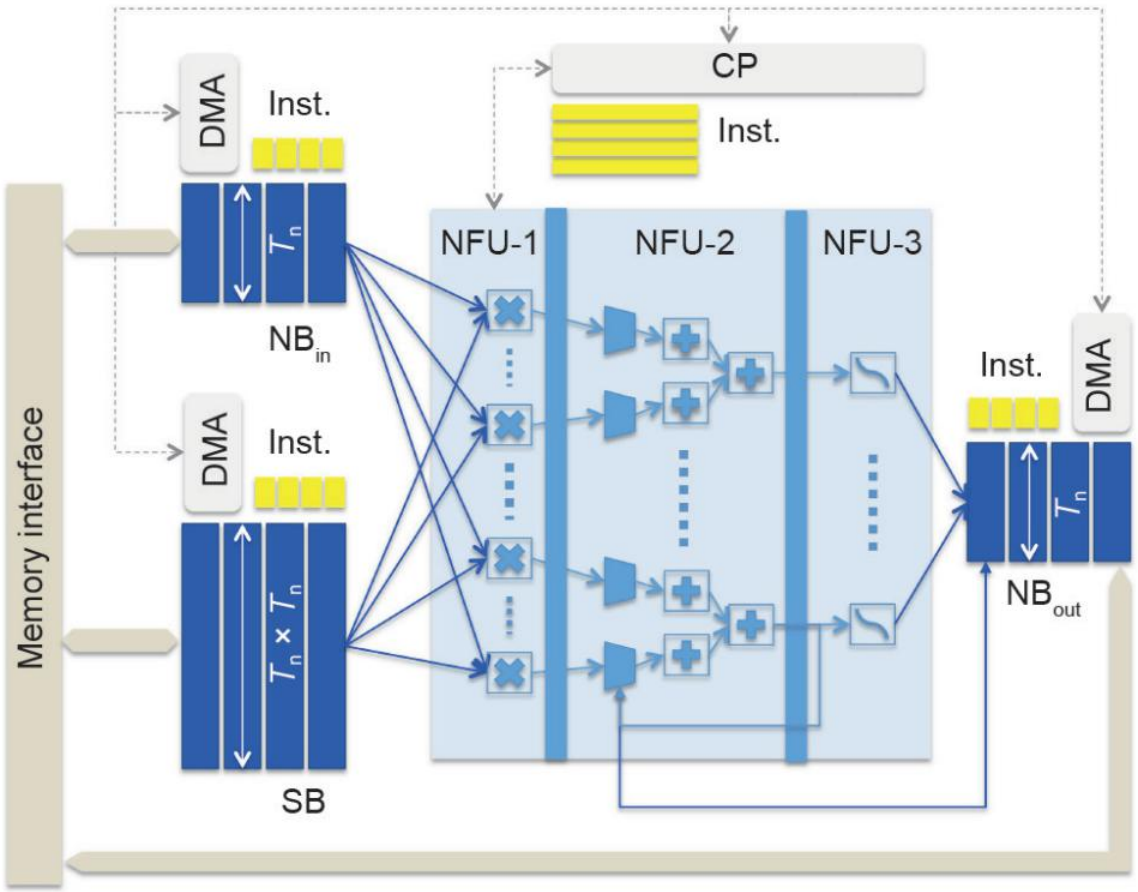

图 6 DianNao 架构

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 [4] 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 [5] 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 [6] 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。

除了独立的加速器之外，电脑系列还提出了一种称为 Cambricon [7] 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。

4.2 TPU

TPU 在工业界取得了巨大的成功，Google 在 2017 年发布了第一篇 TPU 论文（TPU1）[8]，其架构如图 7 所示。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 中，Google 宣布了其云 TPU（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。

TPU 被设计成为一个位于 PCIe 上的协同处理器。为了简化硬件设计和调试，TPU 不具备从内存中取指令的能力，相反主服务器将 TPU 指令送到片上存储当中。因此，TPU 的设计初衷更接近于 FPU（floating-point unit）而不是 GPU。

TPU 指令通过 PCI 总线传递到 instr buffer 中。TPU 的内部模块有 256byte 宽的数据通路连接。Matrix Multiply Unit 是 TPU 的核心，它由 256*256 个 Multiply Add Components（MACs）组成，每个 MAC 可以执行 8bits 整型相乘相加操作。16bits 计算结果存储在由 32bit 单元组成的 4MB 累加器（4K*256*32b Accumulators）中。MMU 每个周期产生一次含 256 个单元的部分和。MMU 存储一个 64KB 的权重 tile。

当使用 8bits 权重和 16bits 激活函数操作时，MMU 以一半的速度执行。当两者都是 16bit 时，计算速度为四分之一。每周期可以读或写 256 个数值，并做矩阵乘操作或是卷积操作。MMU 主要为高密度矩阵而设计，针对稀疏矩阵还有待提升。

MMU 从 on-chip Weight FIFO 获取数据，FIFO 的数据从 off-chip DRAM Weight Memory（推断为只读 Weight）。FIFO 深度为 4 个 tile。计算中间值存储在 24MB 的 on-chip Unified Buffer（UB）当中，同时 UB 可以作为 MMU 的输入。UB 与主存通过 DMA 传输数据。

UB 大小占芯片面积约为三分之一，MMU 大小约为四分之一。片上内存设计尽可能大以使计算本地化，在普通操作当中没有 DRAM 数据载入，减少对内存带宽的要求。

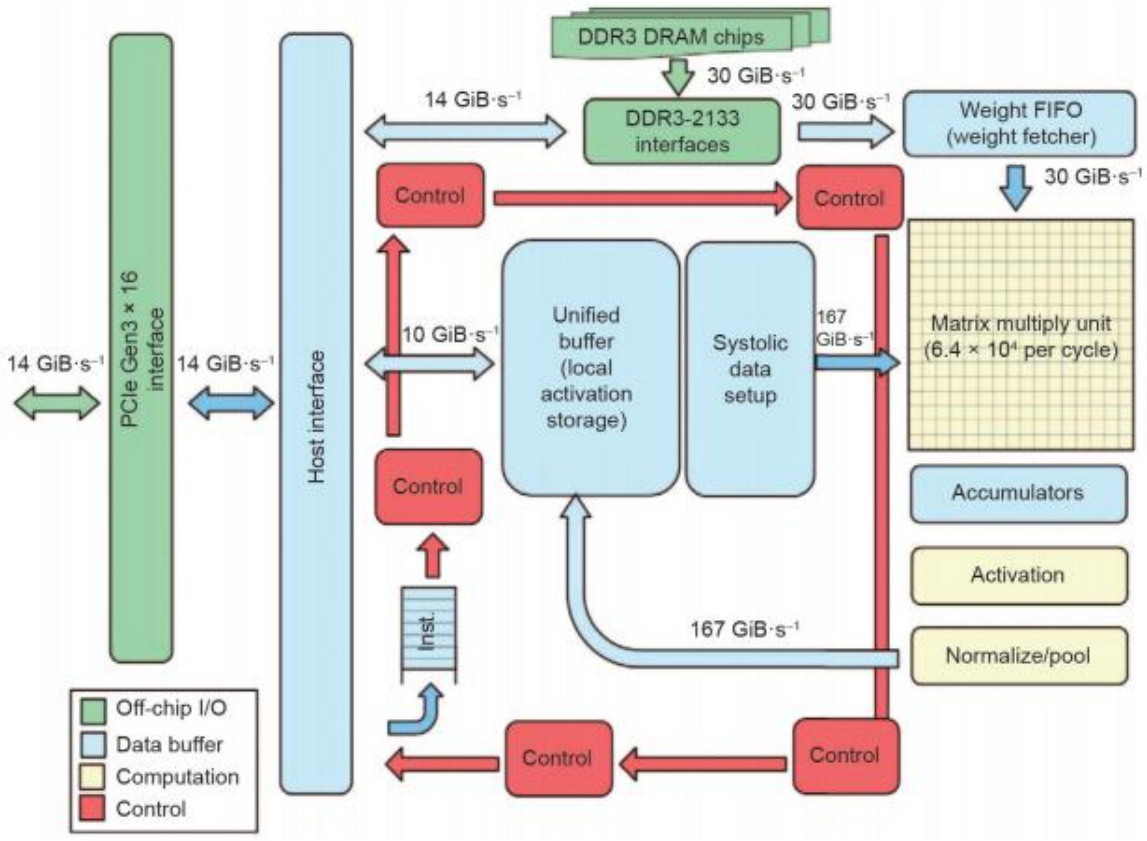

图 7 TPU 架构

5 总结

5.1 报告总结

虽然只是一篇迷你综述，但撰写本报告几乎占据了半周所有的课余时间。前期的调研工作量与正式撰写的工作量不相上下。在大致浏览了 ISCA 和 HPCA 近年来的一些论文后，我了解到针对人工智能领域的特定的硬件加速，是近几年计算机体系结构方面的重点工作，论文量也较多，于是确定了报告的主题。

文章主要介绍了两种片上加速和两个系列的独立加速器，在阅读这些论文的初期，由于能力和背景知识有限，陷入了对于细节的纠缠，报告的撰写进度非常缓慢。在学习了“阅读论文的三遍论”之后，我放弃了对实现细节的把握，着重理解大致的原理和硬件架构，对其设计思想和应用场景进行描述。虽然最后对这些加速器的深层原理还不是很理解，但对于通过运算和存储两方面对神经网络进行加速这一领域的认知程度仍然有了较大的提升。

5.2 课程总结

在完成了期末考试和实验报告之后，本篇综述的完成意味着计算机体系结构课程的结束。在上半学期的专业课中，体系结构是“最硬”的一门，学到的知识也是最庞杂的。本门课程使我对计算机硬件层面的认知在组成原理的基础上大大拔高了一层，使我更加深刻地理解了多周期、流水线、指令调度、缓存等技术。李老师在授课时的热情、对前沿技术动向的关注、对相关知识的补充提示以及实验时给我的帮助（重新编译模拟工具的指令集架构），让我相信他对体系结构这一领域的研究之深入以及对计算机科学的情怀。

由于课程的课时不多，加上体系结构知识面的庞大，我对许多其方面的学习仍然一知半解，例如存储器、IO、多级缓存的具体细节，等等。如何在紧张的学时中更好地、更系统地安排教学计划和进度，是我认为教学团队需要进一步优化的地方，也希望自己能在以后的学习中，对这些遗漏的知识进行补全和深入学习。

参考文献

[1] Esmaeilzadeh H, Sampson A, Ceze L, Burger D. Neural acceleration for general-purpose approximate programs. Commun ACM 2014;58(1):105–15.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, et al. Indatacenter 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.