机器学习加速器——以 DianNao 系列为例

学号:161220093, 作者姓名: 缪源清, 邮箱:myuanqing@outlook.com

2019 年 6 月 18 日

1 Introduction

DianNao 系列均是面向机器学习研究的加速器。DianNao 系列共有 4 个加速器，按发布时间依次为：DianNao(神经网络加速器，发表在 ASPLOS-2014)，DaDianNao(DianNao 的多核版本，发表在 MICRO-2014)，ShiDianNao(视觉处理加速器，发表在 ISCA-2015),PuDianNao(机器学习加速器，支持 7 种机器学习算法，发表在 ASPLOS-2015)

2 Background

DianNao 系列是面向机器学习的，因此有必要在此对其背景进行介绍.

2.1 Neural Network

神经网络的发展之路是为曲折: NN 在上世纪八九十年代名噪一时，却随着支持向量机 (Support Vector Machine) 的流行淡出人们的视野. 然而自 2006 年起，神经网络其下的两个分支：深度神经网络 (Deep Neural Network，即 **DNN**) 与卷积神经网络 (Convolutional Neural Network, 即 **CNN**) 重新使人们将目光转向了神经网络.

2.1.1 General structure

DNN 与 CNN 形式有所不同，但可以定义出一套通用的算法框架. 以下是一些定义.

层 (*layers*): layers 是顺序执行的，因此它们可以被独立地考虑并优化. 主要有三种 layers：卷积层 (*convolutional layer*), 池化层 (*pooling layer*/*sub-sampling layer*) 和分类层 (*classifier layer*). 前二者数目多，classifier layer 位于网络的顶端，仅一或几层.

特征图 (*feature map*): 每个 layer 包含若干 feature maps. Feature map 有 *input feature map* 与 *output feature map* 之分.

2.1.2 Convolutional Layer

卷积层的作用是对输入数据 (前一层的数据) 进行局部的过滤. 因此输入和输出的 feature map 的是局部相连而不是全相连的. 输入是一张图片时，卷积是一个二维变换，变换的单位对象是属于输入层的一个 window 大小的数据. 变换的操作数还包括 window 大小的 *kernel values*. kernel values 是输入层和输出层间的权值. kernel 可以是 3 维的，此种情况下有多个 input feature maps. 某些情况下连接是稀疏的，即不是每个 input feature maps 都会被用到. 有些时候，input layer 的所有 window 共享一组 kernel values，比如在 CNN 模型中; 有时则各异，比如在 DNN 中，这时称它们是 *private* kernel.

2.1.3 Pooling Layer

池化层的作用是将一组相邻的输入数据的信息集中起来. 比如, 在图像中, 只保留一个 window 中最显著的特征. 池化的副作用是减少了 feature map 的维度.

2.1.4 Classifier Layer

分类层通常聚合所有的 feature map，因此对分类层的讨论中不会出现 feature map.

2.1.5 Others

DaDianNao 中提到了局部响应正则层 (*Local response normalization layers*), 它实现了不同 feature map 相同位置的神经元的竞争.

3 DianNao

DianNao 实现了快速的输入处理 (前向的)，而不是训练 (向后反馈的). 这是基于技术和市场的考虑. 技术层面上，*off-line* 的算法可以满足很多应用; 市场层面上，*end users* 占据很大的市场，他们的需求是快速有效的前向网络.

3.1 Accelerator for Neural Networks

设想一种朴素的神经网络加速器: 所有的神经元和突触都排布在硬件中，存储仅在输入和存储结果的时候被访问. 尽管这样的神经网路有可能达到最佳的能量利用率，它并不是可规模化的. 面积、能量、延迟都随着神经元数目的上升呈二次方增长. 因此目前不能采取此方式

加速器主要由以下这些部分组成: 输入神经元的输入缓存 (**NBin**), 输出神经元的输出缓存 (**NBout**), 突触权值的缓存 (**SB**), 神经功能单元 (**NFU**), 控制逻辑 (**CP**).

3.2 NFU

不同类型的 layer 在 NFU 可能经过 2 个或 3 个阶段. Classfier layers 需要进行这些计算（每种占一个阶段）：突触与输入相乘，累加乘积，计算 sigmoid; convolutional layers 经历的阶段与 classifier layers 相同，但是最后一阶段可能进行不一样的计算 (可能是其它非线性函数); pooling layer 不进行乘法计算，没有最后一阶段.

这些阶段被流水化了. 但是流水线是交错式的: 前两个阶段行为如同正常的流水线，但是第三阶段仅在所有的加法操作都完成后才活跃.

DianNao 没有采用 32-bit 的浮点运算部件，而是采用了 16-bit 的定点算数部件. 根据调研，后者对于准确度的影响不大.

3.3 Storage

存储可以通过 cache 和 scratchpad 方式实现. 前者主要应用在通用设备中，但它是一种次优的选择，因为 cache 的访问有 overhead(tag check, associativity, line size, speculative read, etc) 并且有 cache conflicts. 一个有效的替代是应用在 VLIW 处理器中的 scratchpad，但是它很难编译. 然而在一个专用加速器中，scratchpad 既可以有效地存储，又能简单有效地利用局部性，因为只有少量的的算法需要被手动改编.

3.3.1 Split Buffer

前面已经提到, 存储被分为了三个部分: an input buffer(**NBin**), an output buffer(**NBout**) and a synapse buffer(**SB**).
划分的好处有:
1. 将 SRAM 裁成合适的读写宽度. NBin、NBout 与 SB 的宽度不同. 将存储裁成专门的结构可以使每个读请求达到最优的时间和能量效率.
2. 避免冲突. 如果使用高度相连的 cache，那么将很耗电. 对存储的划分和对局部性的确切认知可以完全消除数据冲突. 这可以使得存储尽可能小以省钱节电.

3.3.2 Exploiting the locality of inputs and synapses

1.DMA. DianNao 在三个 buffer 中都实现了 DMA. DMA 请求被以指令的形式发射到 NBin 中. 这些请求都被缓存到每个 buffer 的独立的 FIFO 队列中, 只要 DMA 发射完前面的请求，后面的可以立刻发射. FIFO 使得发往 buffer 的请求和 NFU 得以解耦. 因此 DMA 请求可以被预取，只要有足够的 buffer 容量.

2. Rotating NBin buffer for temporal reuse of input neurons. 将 NBin 实现为一个环状 buffer 可以使得输入层被重用. 实际中，rotation 的实现方式是改变寄存器的索引.

3. Local transpose in NBin for pooling layers. 对 K_x,K_y,N_i 大小的 kernel，memory 的访问以 N_i(即 feature map 的标号) 为最内部的维度时更有效. 然而 pooling layer 的运算是逐 feature map 进行的, 连续存储时更有效. 解决办法是存储到 buffer 前进行一次转置. 使得 N_i 成为最外部的维度.

3.3.3 Exploiting the locality of outputs

1.Dedicated register. 由于数据的传输消耗很多能量，因此在 NFU 的第二级流水线引入 dedicated register，以存储部分和.

2.Circular buffer. 将部分和暂存到 NBout 中，减少与 memory 的交互.

3.4 Control

控制的一种实现方式是将三种目标层硬连线. 但是 DianNao 采用的方式是控制指令 (*control instructions*), 这样可以探索不同的 layer 的实现方式，并且更为灵活. 指令存储在与 CP 相连的 SRAM 中. 由于只有三种类型的代码，因此使用一个编译器是大材小用，DianNao 为三种层实现了三个专用代码生成器.

4 DaDianNao

4.1 The GPU Option

尽管 GPU 的加速明显，它有许多限制. 其一是面积: 硬件操作器的数目和保持通用性的需要都限制住了它; 其二是总的运行时间，对于微秒量级的服务如网页服务或工业应用来说，在 GPU 上的运行时间是不可忍受的; 其三是能量效率，GPU 很耗电.

4.2 The Accelerator Option

DianNao 大大减小了面积，但它的主要限制在于 memory 的带宽要求: DNN 的 convolutional layers 和 CNN、DNN 的 classifier layer. 此外，片外的访存使得总耗电增加了 10 倍左右.

4.3 A Machine-Learning Supercomputer

从上面可以看出，加速器的两个主要问题是 memory storage(for reuse) 和 bandwidth requirement(for fetching).
设计原则是:
1. 建立这样的架构: 突触被存储在靠近将要使用它们的神经元附近，以减小数据移动，节约时间和能量. 架构是完全分布式的，因此没有主存.

2. 建立非对称架构：每个节点的面积更依赖与存储而不是计算.
3. 传递神经元的值而不是突触的值. 因为在前向层中，前者的数目比后者少若干数量级，因此需要相对更少的片间带宽.
4. 将局部存储分割成若干 tiles，以提高内部带宽.

总体的架构是: 有一组 nodes，每个 chip 一个 node，它们以 classic mesh 的拓扑结构相连. 每个 node 都包含大量的存储.

4.3.1 Node

1.*Synapses Close to Neurons.* 使突触的存储靠近神经元并且大量地布置此结构. 这样做有两个目的: 其一，架构的目的是推断和训练. 在推断中，前一层的神经元是后一层的计算的输入; 在训练中，神经元是先前向传播再后向传播的. 鉴于有太多的突触，因此移动神经元的输出而不是突触才是合理的; 其二，使突触靠近计算单元使得数据的传输有低耗电、低延迟的特点，并且可以保有高的内部带宽.
存储采用了 eDram，它有更高的存储密度. 如果 NFU 不再被带宽限制，那么有可能增大它的尺寸以处理更多的神经元. 但是 eDram 也有几个缺点: 比 SRAM 更高的延迟，破坏性读和周期性刷新. 为了弥补这些缺点，DaDianNao 将 eDram 划分成 4 个 bank，并将突触行散落到 4 个 bank 中.

2.*High Internal Bandwidth.* 为了避免拥塞，采取了 tile-based 的设计. 输出神经元被散布到不同的 tile 中，因此每个 NFU 可以同时地处理 16 个输入神经元.

3.*Configurability(Layers, Inference vs. Training)* NFU 被分解成若干的硬件模块：adder block，multiplier block, max block and transfer block. 每个硬件模块都可以被设计成允许 16-bit 的运算器聚合成 32-bit 的运算器的形式. 尽管对于推理，16-bit 的运算器性能足够，但是对训练来说，它们可能会降低准确度、增加收敛时间.

4.3.2 Interconnect

由于神经元是唯一需要被传输的数据，并且它们被重复性很高地复用，因此，通信的数量几乎不是瓶颈. 因此没有专门设计互联电路，而是使用了商业通信器件. 拓扑采用了 mesh 结构.

5 ShiDianNao

ShiDianNao 关注的是图像应用，这是在识别和挖掘类应用中最重要的分支. 这些应用使用的神经网络是卷积神经网络 (CNN). 卷积神经网络的特征是权值被许多神经元共享，因此可以减少神经网络存储的面积. 这条性质使得可以将一个 CNN 映射到一个 SRAM 中，消除了所有为访问权值而进行的 DRAM 访存操作. 将此加速器防止在图像传感器附近可以消除所有的

DRAM 访存.

文中提到，尽管目前在晶体管密度方面已经取得了惊人的成就，实际的 CNN 还是超出了纯空间实现的能力，因此需要对 CNN 做时间的划分，串行地将划分映射到物理计算结构上.

5.1 Accelerator Architecture: Computation

ShiDianNao 新增了两个器件: an arithmetic unit(**ALU**), a buffer and a decoder for instructions(**IB**)

5.1.1 Neural Functional Unit

DianNao 的功能部件在此应用场景下不很有效. 因为它将 2 维的 feature map 当作 1 维的向量处理，不能有效地利用 2 维数据的局部性. 作为对比，ShiDianNao 的 NFU 是一个 2 维的 mesh 结构，即 Processing Elements(**PEs**)，可以适应 2 维 feature map 的拓扑.
1.Processing elements. 每个周期，每个 PE 可以为 convolutional layer, classifier layer, normalization layer 执行一个乘加操作，或为一个 average pooling layer 执行一个加法操作，或为一个 max pooling layer 执行一次比较. PE 单元有三个输入: 控制信号，从 SB 读突触，读神经元; 有两个输出: 向 NBin/NBout 写结果，向相邻的 PE 传递神经元.
2.Inter-PE data propagation. 输入的 window 之间有交叠，因此可以重用.PE 可以向它下面或左面的邻居传送输入神经元. 每个 PE 中有两个 FIFO 队列: FIFO-H 和 FIFO-V. FIFO-H 存放的数据来自 NBin/NBout 或是右邻居 PE，FIFO-V 存放的数据来自 NBin/NBout 或是上邻居 PE. 这样可以减少带宽需求.

5.1.2 Arithmetic Logic Unit

NFU 并没有包含 CNN 所有的计算原语，因此需要一个轻量级的 ALU 实现 PE. 在 ALU 中实现了 16-bit 的定点运算器和非线性激活函数.

5.2 Accelerator Achitecture: Storage

使用片上的 SRAM 同时存储 CNN 所有的数据和指令. 因为近期的研究表明 CNN 只用有限的参数就可以达到高的识别准确度.

5.3 Accelertor Achitecture: Control

5.3.1 Buffer Controllers

NB buffer 的读取有六种形式，对应某一/几层的某个阶段. 控制指令是采用两层的层次型有限状态机 (HFSM) 描述的.

6 PuDianNao

6.1 ML Techniques

PuDianNao 支持 7 中机器学习模型, 它们分别是 *k-Nearest Neighbors*, *k-Means*, *Deep Neural Network*, *Linear Regression*, *Support Vector Machine*, *Naive Bayes*, *Classification Tree*. 模型具体的细节可以阅读相关文章, 在此不再赘述.

总结它们的共通之处:1. 一些关键的计算原语. 如距离计算、点乘、计数和一些非线性函数等. 这些为加速器的运算单元的实现做了提示. 2.tiling 可以在 k-NN,k_Means,DNN,LR 和 SVM 等模型中利用数据的局部性, 但是对 NB 和 CT 并没有帮助. 因此在加速器中实现了三个片上的 buffer, 每个 buffer 存储具有相似的重用半径的变量.

6.2 Accelerator Architecture

PuDianNao 有多个功能部件 (**Functional Unit**s), 三个数据缓存 (**HotBuf**, **ColdBuf** 和 **OutputBuf**), 一个指令缓存 (**InstBuf**), 一个控制模块以及一个 DMA.

6.2.1 Functional Units

FU 是 PuDianNao 的基础运算部件,由两部分组成: Machine Learning functional Unit(**MLU**) 和 Arithmetic Logic Unit(**ALU**).

Machine Learning Unit: MLU 支持几种在代表性的 ML 技术中流行的计算原语. 包括点乘, 距离计算, 计数, 排序, 非线性函数等. MLU 被分成 6 级流水线: Counter, Adder, Multiplier, Adder tree, Acc 和 Misc.

Arithmetic Logic Unit: 有些 ML 技术的运算不能被 MLU 支持, 因此引入 ALU 提高这些运算的速度.

6.2.2 On-Chip Data Buffers

PuDianNao 上有三个独立的片上数据缓存. HotBuf 存放是重用距离较短的输入数据. ColdBuf 存放的是重用距离相对较长的输入数据, OutputBuf 存放的是输出数据或临时结果.

7 Summary

以上是对 DianNao 系列的简单阐述. 可以得出以下几点结论:

1. 设计目标. 加速器 Accelerator 与 GPU 相比, 通用性差, 但是功耗小. 因此应发挥优势, 尽量减小功耗和面积.

2. 访存往往是瓶颈. DianNao 系列都注意到了这一点, 采用 DMA 技术、Buffer、FIFO 队列并充分地利用数据的局部性、重用性.

3. 计算单元的精度. DianNao 系列的计算单元是 16-bit 的, 因为在神经网络中, 计算的准确度

对于结果的准确度影响不大. 若有 32-bit 的要求，则可以将多个 16-bit 拼接成 32-bit.
4. 精细化存储. DianNao 系列为不同的 buffer 设计不同的标准，这可以降低存储的功耗. 在专用芯片中这是非常可取的.

DianNao 是开山之作，DaDianNao 则瞄准了更大规模的神经网络，ShiDianNao 关注图像处理向专用化发展，PuDianNao 集成多种机器学习模型，向通用化发展.

8 References

[1] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengy- ong Wu, Yunji Chen, and Olivier Temam. 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*, pages 269–284. ACM, 2014.

[2]Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. Dadiannao: A machine-learning supercomputer. *In Proceedings of the 47th IEEE/ACM International Symposium on Microarchitecture (MICRO' 14)*, pages 1–14. IEEE, 2014.

[3]Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen and Olivier Temam. ShiDianNao: Shifting vision processing closer to the sensor. In 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA). 92–104.

[4]Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Teman, Xiaobing Feng, Xuehai Zhou, Yunji Chen. PuDianNao: A polyvalent machine learning accelerator. In ACM SIGARCH Computer Architecture News (Vol. 43, No. 1, pp. 369-381). ACM.