GPU 中卷积神经网络加速单元的设计

张炜
（上海兆芯集成电路有限公司，上海 201203）

摘要：GPU 已经被广泛应用于卷积神经网络加速，而传统的 GPU 执行单元主要适用于对 3D 图形渲染进行加速，其性能功耗比与神经网络专用加速芯片有一定的差距。针对卷积神经网络计算和数据的特点，在充分利用 GPU 现有计算单元的基础上，提出了新的稀疏矩阵加速单元的设计方案，解决了 GPU 加速神经网络低性能功耗比的问题。

关键词：集成电路设计；图形处理器；GPU；卷积神经网络；稀疏矩阵。

中图分类号：TN402　文章编号：1674-2583(2019)08-0005-03
DOI：10.19339/j.issn.1674-2583.2019.08.002
中文引用格式：张炜.GPU中卷积神经网络加速单元的设计[J].集成电路应用, 2019, 36(08): 5-7.

CNN Accelerate Unit Design in GPU

ZHANG Wei
(Shanghai Zhaoxin Semiconductor Co.,Ltd, Shanghai 201203, China.)

Abstract — GPU has been widely used in neural network acceleration, and the traditional GPU execution unit is mainly suitable for accelerating 3D graphics rendering, and its performance and power consumption ratio is not as good as that of neural network dedicated acceleration chip. Aiming at the characteristics of neural network calculation and data, on the basis of making full use of the existing computing unit of GPU, a new design scheme of sparse matrix acceleration unit is proposed, which solves the problem of low performance-to-power ratio of GPU when accelerating neural network.

Index Terms — IC design, graphics processor unit, display driver, GPU, CNN, sparse matrix.

1 引言

随着卷积神经网络（Convolutional Neural Network，CNN）的应用和流行，各种加速算法以及对应的硬件加速器的设计层出不穷。目前流行的硬件加速器设计方案包括：GPU、ASIC 和 FPGA。而 GPU 以其自身已经成熟的并行计算硬件架构和软件生态取得了先发优势。以英伟达为代表的基于 GPU 的 CNN 加速器解决方案迅速占领了世界。GPU 的设计目的是为了加速图形渲染和通用计算，而 CNN 固有的并行计算特性完美的适用于用 GPU 加速。但是，CNN 的加速算法只需要用到 GPU 所有功能的一小部分，再加上 CNN 的推理对数据的精度要求不敏感，这就使得用 GPU 加速 CNN 的性能功耗比与专用 CNN 加速器相比没有优势。基于此，GPU 架构设计师在 GPU 中加入了专门用于加速矩阵运算的硬件单元，大幅提高了其性能功耗比[1,2]。本文基于已设计好的CNN加速单元，针对CNN数据的稀疏性做了进一步的改进，使得 GPU 在不增加额外矩阵运算单元的基础上，性能功耗比进一步改善。

2 神经网络结构和数据的特点

2.1 CNN结构

AlexNet 是 CNN 中非常有代表性的网络。从图 1 中能够看到的网络层有：Conv、ReLU、Norm 和 Pool。（1）Conv 被称为卷积层，它是 CNN 各层中计算量最大的。其计算量占到整个网络计算量的 90% 以上。而 CNN 加速器最重要的目标就是对这部分进行加速。（2）ReLU 被称为激活函数，它是非线性函数。当其输入小于 0 时，ReLU 的输出为 0；当其输入大于 0 时，ReLU 的输出等于输入。（3）Norm 一般是 Batch Normalization，在推理的时候，其能被合并入 Conv 层。（4）Pool 是池化层，它有三个作用：① 减少计算量；② 增大感受野；③ 增加平移不变性。

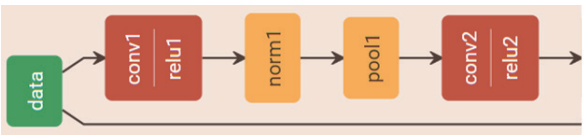

图 1 AlexNet 的一部分

2.2 特征图和权重数据

CNN 中各层的输出被称为特征图，而卷积层所使用的卷积核被称为权重。它们有各自的特点。

2.2.1 特征图数据

图 1 中的 ReLU 层的特性决定了在计算完之后，输出特征图中有很多的 0。因此，它具有很强的稀疏性。而在整个网络执行的过程中，会不断地经过 ReLU 层。经过统计发现，卷积层的输入特征图就具有了很强的稀疏性。

基金项目：工业和信息化部国家核高基（核心电子器件、高端通用芯片及基础软件产品）专项基金（2014ZX01029101）。
作者简介：张炜，上海兆芯集成电路有限公司，研究方向：集成电路设计。
收稿日期：2019-05-24，修回日期：2019-07-12。

表 1 和表 2 分别统计了 AlexNet 和 ResNet50 特征图的稀疏率，表中没有逐层 Layer 统计，而是把稀疏率相似层合并在一起给出总的稀疏率。它们表明了稀疏性在 CNN 中是普遍存在，并且越是后面的层，稀疏率越高。

2.2.2 权重数据

经过训练后的权重不具有稀疏性。但是，它具有自身的特点。其中，最重要的特点是：网络参数

表 1 AlexNet 特征图稀疏率统计结果

Layer ID	sparse_ratio
0	55.46%
1	82.81%
2	66.56%
3	72.56%
4	91.83%

表 2 ResNet50 特征图稀疏率统计结果

Layer ID	sparse_ratio
0	31.47%
1~9	42.12%
10~20	53.79%
21~39	64.05%
40~48	79.63%

存在很大的冗余，很多参数可以被修剪掉。网络的修剪在压缩 CNN 模型的时候被广泛研究。很早之前，网络的修剪就被证明为一个减少网络复杂度和过拟合的一个有效方法。而 Han Song 的论文[3]说明了可以在不影响准确度的前提下修剪 CNN 模型。表 3 是基于其算法统计的结果，它充分说明了经过修剪之后，ResNet50 具有很高的稀疏率。

基于 Han Song 的方法，经过网络修剪，权重的分布是没有规律的，不适合设计简单的硬件进行利用。所以，Huizi Mao提出基于整个卷积核的网络修剪方法[4]。这种方法在牺牲一定的稀疏率的情况下，得到了更加规则的稀疏性（表 4）。

表 3 ResNet50 网络修剪后的精度变化和稀疏率

Model	Sparsity	Granularity	Top-5
ResNet-50	40%	Fine-Pruning	92.34%

表 4 ResNet50 基于卷积核的网络修剪后的精度变化和稀疏率

Model	Sparsity	Granularity	Top-5
ResNet-50	40%	Kernel-Pruning	92.07%

3 GPU 加速卷积层

3.1 GPU 执行单元加速 CNN 的代价

GPU 加速卷积层的算法有很多种，包括 Direct、GEMM、Winograd 和 FFT。这里面除了 Direct，其他三类都可以最终归结为矩阵的运算。因此，无论是英伟达还是谷歌，其设计的硬件加速器都是针对矩阵运算加速的。

众所周知，GPU 的执行单元是 SIMD 方式运行的。即，多个计算共用一条指令。而典型的设计就是 32 个 Data 共用一个指令。而且，更重要的是这 32 个 Data 被定义的精度是单精度浮点数。为了充分利用这些计算单元，提高 ALU 的计算能力，目前半精度浮点计算也得到了很好的支持。而半精度浮点数的性能是单精度的 2 倍，在原来的基础上支持它不需要花费多少代价。

无论如何，GPU 这种对浮点运算单元的支持已经大大超出了 CNN 做推理的精度需求。据研究，实现 CNN 推理且不影响最终的预测精度，8-bit 整型数就能满足要求。而实现 8-bit 整型矩阵之间的乘法运算所需要的逻辑门数目要比同样的浮点运算单元小很多。因此，独立出矩阵加速单元成为设计者的共识。比如，英伟达称之为 Tensor Core。

3.2 卷积神经网络加速算法

上节提到了矩阵加速单元以及卷积层可以转化为矩阵运算。而 GEMM 是最常用的算法。

图 2 可以清楚地说明对输入特征图的卷积计算能够被变换为卷积核与一个展开的特征图矩阵之间的乘积。而矩阵运算更便于设计专用的矩阵加速单元进行加速。

4. 稀疏矩阵加速单元的设计

a0	a1	a2	a3	a4
b0	b1	b2	b3	b4
c0	c1	c2	c3	c4
d0	d1	d2	d3	d4

a0	a1	a2	b0	b1	b2
a1	a2	a3	b1	b2	b3
a2	a3	a4	b2	b3	b4
b0	b1	b2	c0	c1	c2
b1	b2	b3	c1	c2	c3
b2	b3	b4	c2	c3	c4
c0	c1	c2	d0	d1	d2
c1	c2	c3	d1	d2	d3
c2	c3	c4	d2	d3	d4

图 2 卷积核为 3×3 的展开矩阵

4.1 整体架构设计

图 3 是整体架构框图。其中，线程调度单元负责动态调度线程填充浮点元算单元和矩阵运算单元，其需要在多个线程之间调度，选择操作数已经准备好的线程指令发到计算单元；浮点运算单元是 GPU 里面的单精度浮点运算单元，它也支持整数计算；矩阵运算单元是用来加速整数矩阵运算的单元；寄存器文件用来存储计算单元输出的结果和数据读写单元从外部存储器加载进来的数据；断言寄存器用来指示 SIMD 指令中哪些操作数不需要执行；读写控制单元是负责把计算的结果写入存储器以及从存储器加载数据进寄存器；稀疏检测单元为每个矩阵运算指令检测稀疏性，并把检测结果记录在断言寄存器中。

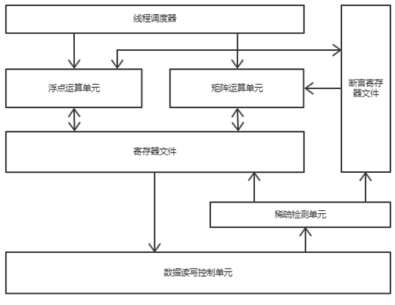

图 3 整体架构

4.2 矩阵加速单元设计

上一节详细介绍了架构图中各个模块的作用。其中，矩阵运算单元和稀疏检测单元是本文的创新点。就如图 3 所示，每个浮点运算单元会搭配一个

矩阵运算单元，在矩阵运算单元中放置了 4 个矩阵加速核心，每个加速核心可以计算出两个 4×4 矩阵的乘积。

图 4 是举证加速单元架构图。其中，MUCTRL 是矩阵加速单元控制器；MU 是 4×4 矩阵加速单元；RF 是寄存器，用来存储矩阵乘法的两个操作数；PRF 是断言寄存器，用来存储稀疏信息；ACC 是累加器，用来存储矩阵运算的中间数据。这个就是现了 D=A×B+C 的功能，其中 C 为中间数据。

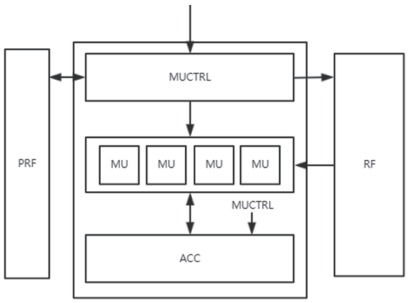

图 4 矩阵加速单元架构图

MU（Matrix Compute Unit）中包括 8-bit 整数乘法器和 16-bit 整数加法器。在跟 ACC 中的数据相加时，使用了 32-bit 的整数加法器。这么做是为了防止数据溢出，影响精度。它的工作过程如下：① MUCTRL 接收线程调度器发送下来的指令。② MUCTRL 解析指令，拿出 PRF 地址去读稀疏掩码。③ MUCTRL 根据稀疏掩码，发送读请求到 RF 和ACC。④ MUCTRL 等数据回来之后，驱动 MU 执行矩阵运算。⑤ MU 计算完之后，把结果写入 ACC。

每个 MU 完成 2 个 4×4 矩阵的乘法，每个 MU 中的加法器的设计采用树形结构。

图 5 给出了一个 MU 中的一个最小单元，每个 MU 有 16 个这种单元，可以输出一个 4×4 的矩阵。其中乘法器是 8-bit 整数乘法器，中间 2 个加法器是 16-bit 加法器，最下面的加法器是 32-bit 整数加法器。

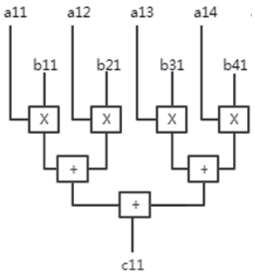

图 5 MU 中的最小单元

4.2.1 稀疏掩码

稀疏掩码是用来确定 MU 需要读取哪些数据，MU 在读取的时候会直接跳过掩码为 0 的数据。掩码的设计如图 6 所示。每个方块就是一个 4×4 矩阵。每个 4×4 用一个 bit 表示，当其中所有数据都为 0 时，则用 0 表示；当其中有任意一个数据不为 0 时，则用 1 表示。

1	0	0	0
0	1	0	0
1	0	1	0
0	0	0	1

图 6 稀疏掩码

图 7 是 2 个 16×16 的矩阵相乘，每个方格是一个 4×4 矩阵。循环顺序为：

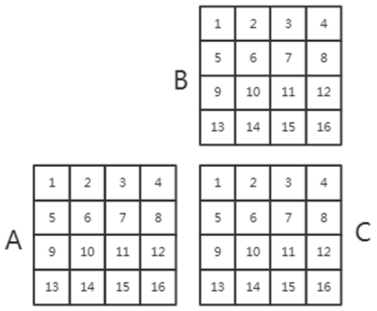

图 7 矩阵相乘

① A1×B1_2_3_4+C1_2_3_4=C1_2_3_4；② A2×B5_6_7_8+C1_2_3_4=C1_2_3_4；③ A3×B9_10_11_12+C1_2_3_4=C1_2_3_4；④ A4×B13_14_15_16+C1_2_3_4=C1_2_3_4。

以上就把 C_1_2_3_4 计算完成了。依据这个顺序，接下去会计算出 C 的所有 4×4。从上面的公式可以看出，A1 或者 B1_2_3_4 有一个为 0，MU 在当前周期就不用读取 A1、B1_2_3_4 和 C1_2_3_4。所以，稀疏性越高，能节省的计算就越多。据统计，ResNet50 可以省略 22% 的矩阵计算。

4.2.2 稀疏检测单元

稀疏检测单元会把从外部存储器中取得的数据按照规定与 0 做比较。比较的粒度是 4×4。这种比较都是基于整数的比较，所以，所费逻辑门数量不多。

5 结语

随着卷积神经网络的流行。各种基于其计算特点的加速器设计层出不穷。本文通过对卷积神网络的研究，在 GPU 中针对性的设计了专门的矩阵运算加速模块，并且充分利用了数据的稀疏性进一步减少了所需要的矩阵运算量，大大提高了硬件加速性能。

参考文献

[1] Tianshi CHEN,Zidong DU,Ninghui SUN,Jia WANG,Chengrong WU,Yunji CHEN,Olivier Temam.DianNao:A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning[J].ACM SIGPLAN Notices 2014,49(4):269-284.

[2] Joseph L.Greathouse,Mayank Daga. Effective Sparse Matrix-Vector Multiplication on GPUs using the CSR Storage Format[C].13th ACIS International Conference,2012.

[3] Song HAN,Huizi MAO,William J.Dally. COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING,TRAINED QUANTIZATION AND HUFFMAN CODING[J].Fiber 56.4(2015):3-7.

[4] Huizi MAO,Song HAN,Jeff Pool,Wenshuo LI,Xingyu LIU,Yu WANG, William J.Dally. Exploring the Regularity of Sparse Structure in Convolutional Neural Networks[J]. Research Gate,2017.