DOI:10.19651/j.cnki.emt.2415345

基于 FPGA 的稀疏卷积神经网络加速器设计*

李 宁 肖 昊
(合肥工业大学微电子学院 合肥 230601)

摘 要:剪枝是一种减少卷积神经网络权重和计算量的有效方法,为 CNN 的高效部署提供了解决方案。但是,剪枝后的稀疏 CNN 中权重的不规则分布使硬件计算单元之间的计算负载各不相同,降低了硬件的计算效率。文章提出一种细粒度的 CNN 模型剪枝方法,该方法根据硬件加速器的架构将整体权重分成若干个局部权重组,并分别对每一组局部权重进行独立剪枝,得到的稀疏 CNN 在加速器上实现了计算负载平衡。此外,设计一种具有高效 PE 结构和稀疏度可配置的稀疏 CNN 加速器并在 FPGA 上实现,该加速器的高效 PE 结构提升了乘法器的吞吐率,同时可配置性使其可灵活地适应不同稀疏度的 CNN 计算。实验结果表明,提出的剪枝算法可将 CNN 的权重参数减少 50%~70%,同时精度损失不到 3%。相比于密集型加速器,提出的加速器最高可实现 3.65 倍的加速比;与其他的稀疏型加速器研究相比,本研究的加速器在硬件效率上提升 28%~167%。

关键词:卷积神经网络;硬件加速器;稀疏计算;FPGA
中图分类号:TP302 **文献标识码**:A **国家标准学科分类代码**:510.4030

Accelerator design of sparse convolutional neural network based on FPGA

Li Ning Xiao Hao
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: Pruning is an effective approach to reduce weight and computation of convolutional neural network, which provides a solution for the efficient implementation of CNN. However, the irregular distribution of weight in the pruned sparse CNN also makes the workloads among the hardware computing units different, which reduces the computing efficiency of the hardware. In this paper, a fine-grained CNN model pruning method is proposed, which divides the overall weight into several local weight groups according to the architecture of the hardware accelerator. Then each group of local weights is pruned independently respectively, and the sparse CNN obtained is workload-balancing on the accelerator. Furthermore, a sparse CNN accelerator with efficient PE and configurable sparsity is designed and implemented on FPGA. The efficient PE improves the throughput of the multiplier, and the configurability makes it flexible to compute CNN with different sparsity. Experimental results show that the presented pruning algorithm can reduce the weight parameters of CNN by 70% and the accuracy loss is less than 3%. Compared to dense accelerator research, the accelerator proposed in this paper achieves up to 3.65x speedup. The accelerator improves the hardware efficiency by 28~167% compared with other sparse accelerators.

Keywords: convolutional neural network; hardware accelerator; sparse computing; FPGA

0 引 言

卷积神经网络(convolutional neural network, CNN)已经成为深度学习领域里最重要的技术之一,其强大的特征提取能力在图像分类、目标检测和语音处理等领域得到了广泛的应用[1-3]。但 CNN 庞大的计算量却为其应用部署带来了挑战,基于中央处理器(central processing unit, CPU)的推理平台因为低并行性而导致较高的延迟,使用高性能图形处理器(graphics processing unit, GPU)计算会产生较大的功耗,因此通用计算设备难以同时满足实时性和低功耗的需求。现场可编程门阵列(field programmable gate array, FPGA)作为一种可编程配置的逻辑器件,能够灵活地实现各种硬件计算架构,在提供高性能的同时又具有较低的功耗,是加速 CNN 的理想选择。缪丹丹等[4]和吴

收稿日期:2024-01-13
*基金项目:国家自然科学基金(61974039)项目资助

宇航等[5]通过数据定点量化、流水线和并行处理的方法设计了基于 FPGA 的 CNN 加速器，具有比 CPU 和 GPU 更优的计算能效比。王习东等[6]基于快速卷积算法设计了 CNN 加速器，有效地减少了乘法计算次数。杜忠文等[7]针对 CNN 加速器中内存带宽瓶颈问题，提出一种基于次级缓存的行重组调度策略，提高了存储器的有效带宽利用率。然而，CNN 中存在大量的冗余参数，对其进行剪枝只会造成极小的精度损失[8-9]。以上的密集型加速器都无法利用剪枝带来的稀疏性来跳过冗余的参数计算，从而降低 CNN 的计算复杂度以提高计算性能。

根据不同的剪枝粒度，一般分为结构化剪枝[10-11]和非结构化剪枝[12-13]。结构化剪枝按照一定规则剪去网络中的特定通道、滤波器、行或列，剪枝后的网络结构具有较好的规则性。非结构化剪枝可以修剪网络中任何位置的参数，被修剪的通常是绝对值较小的参数，剪枝后的模型结构规则性较差。非结构化剪枝因其修剪粒度更细，比结构化剪枝的精度更好且剪枝比更高。然而，非结构化剪枝却造成了不规则的权重分布，即在不同的稀疏卷积核或通道中非零的有效权值数量和比例是不相等的，这就导致了硬件处理单元(processing element，PE)之间的计算负载不一致。因此，小负载的 PE 必须插入空闲周期等待负载更大的 PE 计算完成，降低了部分 PE 的实际计算性能和利用率。

针对剪枝带来的计算负载失衡问题，Lu 等[14]提出了一种切片查找表，将权重与相应的数据切片相匹配，并仅将有效的权重和数据分配给 PE 进行计算。然而，随着滤波器尺寸的增加，实现切片查找表所需要的存储资源也变多。文献[15]和[16]提出将稀疏卷积核中的非零权值重新排列成密集矩阵后再进行计算，以确保每个 PE 的计算负载是相同的，但代价是给数据收集模块带来了额外的复杂性。此外，一部分研究从算法方面优化网络以解决负载不平衡的问题。Li 等[17]提出了一种模式剪枝方案，该方案用四种固定的剪枝格式来保证 PE 间的负载平衡性，但这也使该方法的拓展性较弱。Sun 等[18]对整体的卷积核采用了相同的剪枝率，但这种方法会对高敏感度卷积核造成过度剪枝而使模型精度下降，同时又对低敏感度卷积核的剪枝不足。

针对以上问题，本文首先提出了一种局部权重独立剪枝算法，该算法将模型权重分成若干个局部权重组，并分别对每一组局部权重进行独立剪枝。同时，基于 FPGA 设计了一个具有高效 PE 结构和稀疏度可配置的稀疏 CNN 硬件加速器，通过优化乘法计算和解决权重排列的零填充问题来提高乘法器的吞吐率和效率，同时其可配置性使其能够支持任意稀疏度的卷积计算。

1　软件算法设计

针对非结构化和结构化剪枝所存在的问题，本文提出了一种局部权重独立剪枝(local weight independent pruning，LWIP)算法。在 LWIP 算法中，引入 3 个超参数：M、N 和 T。其中 M 和 N 是与硬件加速器架构相关的两个参数，分别代表加速器在输入和输出通道维度上的计算并行度；T 是剪枝阈值。LWIP 算法的剪枝过程如图 1 所示，图中的 i_x 代表第 x 个输入通道，o_y 代表第 y 个输出通道。为了简单起见，这里将 M 和 N 都设为 2 来解释 LWIP 算法。LWIP 算法首先将尺寸为 $4\times4\times3\times3$ 的权重在输入通道维度上按每 2 个($M=2$)通道分割一次，同理在输出通道维度上按每 2 个($N=2$)分割一次，将整体的权重分成 4 个 $2\times2\times3\times3$ 的局部权重组(local weight group，LWG)，即 LWG_1、LWG_2、LWG_3 和 LWG_4。这样的分组方式结合了加速器的计算架构，将加速器在同一批次计算的权重分到同一组中。分组数量的计算公式为：

$$N(I_c,O_c)=\left\lceil\frac{I_c}{M}\right\rceil\times\left\lceil\frac{O_c}{N}\right\rceil \tag{1}$$

其中，I_c 和 O_c 分别代表输入和输出通道的数目，⌈ ⌉表示对计算结果向上取整。

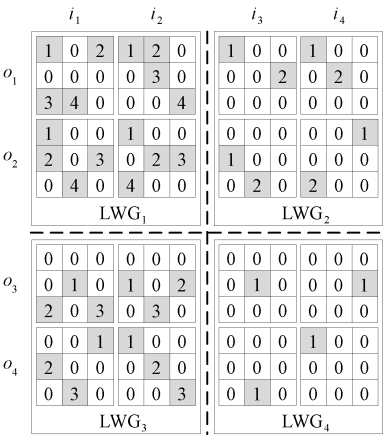

图 1　LWIP 算法

接下来，LWIP 算法对每个 LWG 独立剪枝。根据所设定的剪枝阈值 T，LWIP 算法在 LWG 内部统计每个滤波器使用 T 时的剪枝率。滤波器的剪枝率越高，则其对应的敏感度越低；反之亦然。直接使用统计得到的剪枝率进行剪枝，会造成硬件加速器中不同 PE 之间的计算负载不平衡，因此 LWIP 算法计算单个 LWG 内部所有滤波器的平均剪枝率并将其作为整个 LWG 的剪枝率。这样，在 LWG 内部的每个滤波器将采用相同的剪枝率，使得加速器中同一批计算的所有 PE 的负载总是相等的。不同 LWG 的剪枝操作相互独立，因此在加速器上属于不同计算批次的 LWG 的剪枝率并不相等。从图 1 可以看出，LWIP 算法使 LWG_1、LWG_2、LWG_3 和 LWG_4 中的滤波器保留的计算负载分别为 4、2、3 和 1。这样，在每个 LWG 内部采用了相同的剪枝率，由于 LWG 之间的独立性仍然让整个网络的稀疏度具有随机性。最后，对剪枝得到的稀疏 CNN 模型进行微调和训练，通过对剩余权值的迭代更新来弥补剪枝带来的精度损失。

本文对比了 LWIP 和其他算法的剪枝模型负载分布特点,如表 1 所示。结构化剪枝从整个网络层中移除特定的滤波器、行或列,其负载分布在局部和整个网络层中都是相同的;非结构化剪枝从整层网络权重中移除阈值较小的权重,其在局部和整层的负载分布均是随机的;文献[18]采用了较为固定的剪枝方式,其负载分布与结构化类似;而本文的 LWIP 算法在对 LWG 的剪枝过程中采用了类似结构化剪枝的方式,因此其局部的负载分布是相同的,但是对不同的 LWG 又采用了不同的剪枝率,从而在整层网络中的负载分布是随机的。结构化剪枝和文献[18]的剪枝粒度较粗,造成对高敏感度滤波器的过度剪枝和低敏感度滤波器的剪枝不足;而 LWIP 算法将权重分组并根据局部权重的敏感度选择合适的剪枝率,因此剪枝粒度比结构化剪枝和文献[18]中的方法更细,从而保留了更多重要的权重特征;同时局部负载分布相同的特点又保证了硬件计算单元的负载平衡性。

表 1　不同剪枝方法的负载分布特点

剪枝方法	负载分布
结构化剪枝	局部相同,整层相同
非结构化剪枝	局部随机,整层随机
文献[18]	局部相同,整层相同
LWIP	局部相同,整层随机

考虑到剪枝后的稀疏 CNN 中大量的零权重在访存时会产生无效的带宽,因此本文对稀疏 CNN 的权重进行压缩以降低访存过程中的零权重比例。本文采用"数据+坐标"的组合方式将所有非零权值压缩保存,所有的零权重被全部删除。压缩方法如图 2 所示,每个滤波器内部的数据坐标都从 1 开始,用坐标 0 作为两个滤波器之间的分界线。这样使得权重在存储时的零参数减少,在权重的访存过程中也利用稀疏性特点降低了访存带宽和功耗。

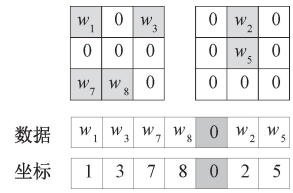

图 2　稀疏权重压缩

2　硬件加速器设计

2.1　硬件架构

本文提出的加速器的顶层硬件架构如图 3 所示,主要包括数据和权重缓冲区、稀疏卷积单元、后处理单元和输出控制器。权重和特征数据默认存储在外部存储器中,计算时加载到片上缓冲区。稀疏卷积单元负责完成对稀疏滤波器的运算,后处理模块对稀疏卷积单元的输出结果进行量化、激活和池化计算,最终输出控制器经过总线将计算结果写到外部存储器中。

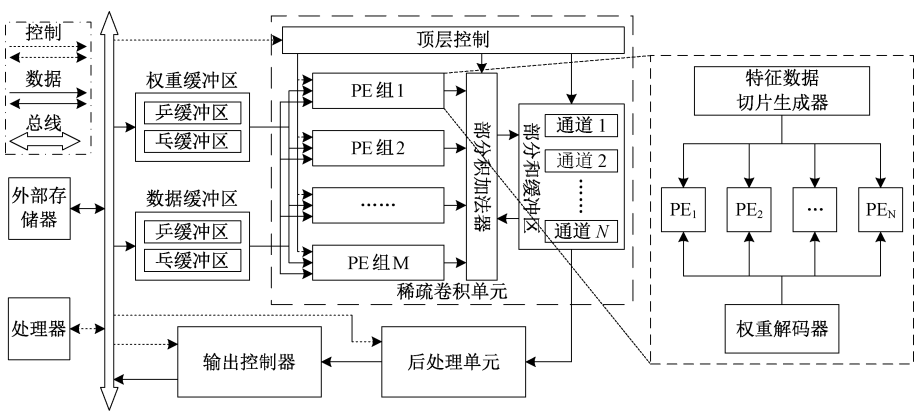

图 3　顶层硬件架构图

2.2　数据流优化

卷积的计算公式如下:

$$\boldsymbol{O}(n,h,w)=\sum_{i=1}^{I_c}\sum_{p=1}^{K_h}\sum_{q=1}^{K_w}\boldsymbol{K}(n,i,p,q)\boldsymbol{I}(i,hs+p,ws+q) \tag{2}$$

其中,$\boldsymbol{I}$ 为输入特征图;$\boldsymbol{K}$ 为卷积核;$\boldsymbol{O}$ 为输出特征图;n、h、w 分别为输出特征图的通道、高和宽;I_c 代表输入特征图的通道数;K_h、K_w 为滤波器的高和宽;s 是卷积步长。

从式(2)可以看出,输入通道为 i 的特征数据将会与不同输出通道的卷积核进行计算,因此输入特征数据可进行复用以供多个卷积核同时运算。记为:

$$\boldsymbol{D}(n,i,h,w)=\sum_{p=1}^{K_h}\sum_{q=1}^{K_w}\boldsymbol{K}(n,i,p,q)\boldsymbol{I}(i,hs+p,ws+q) \tag{3}$$

则式(2)可以改写为:

$$\boldsymbol{O}(n,h,w)=\sum_{i=0}^{I_c}\boldsymbol{D}(n,i,h,w) \tag{4}$$

在式(3)中,如果保持 i 不变,令 n 的取值为 1～N,则可以计算得到 N 个输出部分积:$\boldsymbol{D}(1,i,h,w)$、$\boldsymbol{D}(2,i,h,w)$、…、$\boldsymbol{D}(N,i,h,w)$。此时在计算这 N 个输出通道的部

分积时所使用的输入特征数据相同，因此共用特征数据提高了其片上利用率，减少了特征数据从外部存储器中的访问次数，从而降低了硬件加速器对外部存储器的访存带宽和访存功耗。N 越大，输入特征数据的复用性越高，但是这也需要更多的存储资源来缓存输出结果。

在式(4)中可观察到对于输出通道 n 相同的部分积 $\boldsymbol{D}$ 需要全部相加才能得到最终结果，如果并行计算 M 个相同输出通道的部分积：$\boldsymbol{D}(n,1,h,w)$、$\boldsymbol{D}(n,2,h,w)$、…、$\boldsymbol{D}(n,M,h,w)$，则这 M 个部分积可在全部相加后再存入部分积缓存区。当 M 个输入特征图被并行计算时，每个输入特征图以式(3)的方式被复用并计算 N 个输出通道的部分积，则一共得到 M 组部分积。最后，将 M 组部分积中输出通道相同的计算结果相加后存入缓冲区中，这样优化后使得部分积不需要分别存储在不同的缓存中，仅将加法结果写入一个缓存中即可，从而提高了存储资源的复用性，减小了缓冲区空间。

2.3　处理单元(PE)

文献[4]中的密集型加速器利用 CNN 在边缘推理计算时被量化为 8 位定点数的特点，使 FPGA 中的一个硬件乘法器同时进行两个乘法操作，如式(5)所示。

$$[(w_1 \ll 16)+w_2]\times d=[(w_1\times d)\ll 16]+w_2\times d \tag{5}$$

其中，w_1 和 w_2 表示两个不同滤波器中的权重，d 表示一个特征数据，将 w_1 左移 16 bit 后与 w_2 相加并相乘，则最后从乘积中可根据高低位分离出两个乘法结果。

但是在稀疏型加速器中由于不同滤波器之间的有效权重位置不同，因此同一个特征数据不一定对两个滤波器中的权重都有效，导致式(5)中的乘法器优化方法无法适用于稀疏 CNN 加速器中，降低了乘法器的吞吐率。对此，本文提出基于权重复用和数据切片的稀疏计算方法，该方法在一个乘法器中复用滤波器的权重与两个特征数据切片进行稀疏计算，并得到两个输出结果，每个数据切片的大小与滤波器尺寸相同。本文所提出的 PE 原理图如图 4 所示，首先设计了两个特征数据切片缓存区，分别存储奇数项和偶数项的数据切片，并通过寄存器 Reg1 和 Reg2 来保存数据切片以供乘法器访问。通过路径②将 Reg2 中的数据切片共享给左边的乘法器，因此左边的乘法器同时使用一个权重与两个数据切片进行乘法计算。同理，Reg1 中的数据切片共享给右边的乘法器，并使右边的乘法器也同时完成两次乘法。最后，通过路径③将来自偶数项切片数据的积输出到右边的加法器，将奇数项切片数据的积输出到左边的加法器，所以左右两个加法器的最终输出结果分别是来自奇数项和偶数项的数据切片的计算结果。本文的稀疏计算方法，使稀疏 CNN 加速器中的乘法器也如密集型加速器一样能够同时计算两次乘法。

如图 5(a)所示，Cambricon-X[19] 将相同维度的权重按照每个地址存放 4 个权重数据的方式进行排列，当权重的

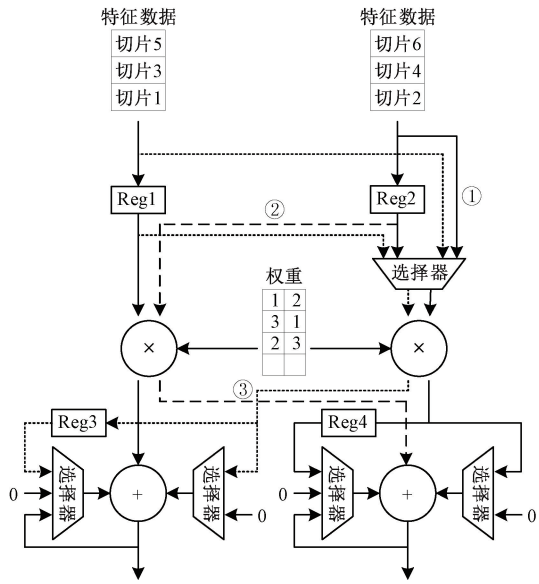

图 4　PE 原理图

数量不是 4 的整数倍时，对最后一行的剩余位置使用零进行填充，这样的数据重组形式又重新引进了非必要的零权值，使部分乘法器仍然参与了零相关的乘法计算。对此，本文提出采用“权重二次排列”方法进行重组。由于本文的 PE 中的乘法器数量为 2，因此每个地址上排列两个权重。如图 5(b)所示，当有效权重数量为奇数时，则在最后一个地址上需要用零进行填充。但当使用“权重二次排列”方法时，则用权重 w_1 代替原来的零，同时将剩余的权重再依次排在后面。当 PE 遍历计算一次所有的权重后，将有两个数据切片参与计算并得到两个结果，但却不存在任何零相关的乘法计算。如图 5(c)所示，当有效权重的数量为偶数时，此时在最后一个地址没有零填充，但为了与奇数同步，仍需从下一个地址再排列一遍所有的非零权重。本文提出的权重排列方法解决了零填充的问题，消除了加速器对任意零权重的访问，有效地提高了乘法器的效率。

当有效权重数量为 3 时，则第 2 个地址上的一个权重需要访问当前的数据切片计算，而另一个权重则需要访问下一个数据切片从而计算下一个结果，上述的 PE 设计显然无法满足此要求。对此，添加一条“直连通路”将下一个数据切片直接送到右边的乘法器中，即图 4 中的路径①。在上述情况下，当前计算切片仍还有一个权重的乘积未相加，所以通过寄存器 Reg3 和 Reg4 将下一个切片的计算结果暂存并在下一个周期参与累加。最后分别在每个乘法器的左边添加一个多选器用以选择累加的数据，同理右边的乘法器用来选择乘积结果是否要累加到当前的结果中。

2.4　稀疏度可配置的稀疏卷积单元

在稀疏卷积单元中一共设计了 M 个 PE 组，每个 PE

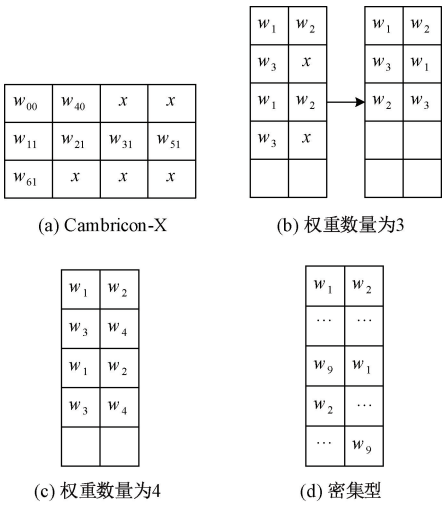

图 5　权重排列方法

组中包含 N 个 PE 且全部利用同一张输入特征图计算,因此稀疏卷积单元可同时计算 M 个输入特征图并行输出 M 组部分积。在 PE 组中还分别设计了特征数据切片生成器和权重解码器,其中前者用来生成特征数据切片并按照奇数和偶数的顺序存入特征数据缓存区中,权重解码器将压缩的权重进行解码并按照“权重二次排列”的方法为每个 PE 提供权重。设计了一个部分积加法器,用以完成 M 组部分积沿着输出通道的相加任务。当输入特征图通道大于 M 时,此时需要进行多批次计算并与对应输出通道的历史计算结果相加,对此本文设计了一个部分和缓冲区,用以缓存 N 个通道每次所得的部分和。

为了适应不同稀疏度的卷积核计算,在稀疏卷积单元中设计了一个顶层控制器,该模块拥有一个高级可扩展互连(advanced extensible interface, AXI)总线接口,处理器通过该接口将权重稀疏度和结构信息配置给控制器。在该模块下有一个与处理器时钟频率相同的寄存器堆用来接收和保存处理器的配置信息,同时处理器可通过读取该模块中的状态寄存器来实时地获取计算状态。稀疏度可配置的特性使稀疏卷积单元不仅能够加速计算稀疏 CNN,同时也可拓展应用于推理密集型 CNN。如图 5(d)所示,对于密集型 CNN 的权重,将所有的权重按照“权重二次排列”的方法进行排列,同时在顶层控制器中将稀疏度配置为 0,这样便可使稀疏卷积单元适用于计算密集型 CNN。因此,可配置特性使硬件加速器的通用性得到增强。

2.5　后处理单元

后处理单元主要负责实现量化、激活和池化计算。激活和池化均采用基于门控的设计,对于不需要做激活或者池化计算的数据,则直接绕过运算单元从选择器的另一端输出。具体电路如图 6 所示,激活使能和池化使能是两个来自处理器的配置信号,它们控制激活和池化单元对量化模块的输出结果执行对应的计算。为了提高并行性,后处理单元的数据并行度等于稀疏计算单元的输出并行度 N,可同步处理 N 个通道的输出特征数据。

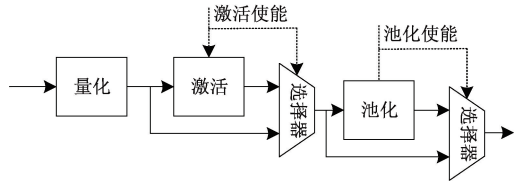

图 6　后处理单元

3　实验与评估

3.1　软件算法评估

本文在 Pytorch 深度学习框架下实现了 LWIP 算法,选用计算复杂度较高的 VGG16 网络在 CIFAR-100 数据集上进行剪枝测试。首先,在 Pytorch 框架下训练全精度的 VGG16 网络;接下来调用 LWIP 方法进行剪枝并得到稀疏网络;最后再对剩余的权值进行迭代更新来弥补剪枝带来的精度损失。分别在不同的剪枝率下统计 LWIP 算法的精度损失变化并与其他的剪枝方法进行对比,结果如表 2 所示。

表 2　不同剪枝方法的精度损失　　%

稀疏度	30	50	70
结构化剪枝	0.6	2.3	4.7
非结构化剪枝	0.1	0.5	1.5
文献[18]	0.3	1.6	3.3
LWIP	0.2	1.2	2.4

由表 2 可以发现 LWIP 算法在 3 种不同的稀疏度下的精度均高于结构化剪枝和文献[18]所提出的剪枝算法,LWIP 采用了比结构化剪枝更细的剪枝粒度,兼顾了局部权重对模型整体精度的影响,使得剪枝后的模型具有更好的精度。同时,LWIP 算法在 70%的稀疏度下的精度损失比非结构化剪枝仅高 0.9%。因此,LWIP 算法的精度介于结构化和非结构化剪枝之间,但局部相同的负载分布却易于硬件实现负载平衡。

此外,本文还通过仿真验证了 LWIP 算法在负载平衡方面的表现。分别使用非结构化剪枝方法和 LWIP 算法对 VGG16 模型进行剪枝,使其稀疏度保持在 70%,并使用 Power-of-Two[20] 量化方法将模型中的 32 位浮点型数据转为 8 位定点型,随后使用加速器进行推理测试。由于结构化剪枝和文献[18]的方法与 LWIP 算法的模型局部负载分布特点相同,因此这里仅对非结构化剪枝和 LWIP 算法作了对比。分别统计硬件加速器推理两种不同的稀疏模型时的 PE 利用率,结果如图 7 所示。经过 LWIP 算法剪枝和优化后的模型在加速器上执行推理时的 PE 利用率更高,这是因为 LWIP 算法消除了 PE 和 PE 组之间的负载不平衡,使 PE 的无效计算和等待周期数量减少,从而有效提

升了硬件的利用率。

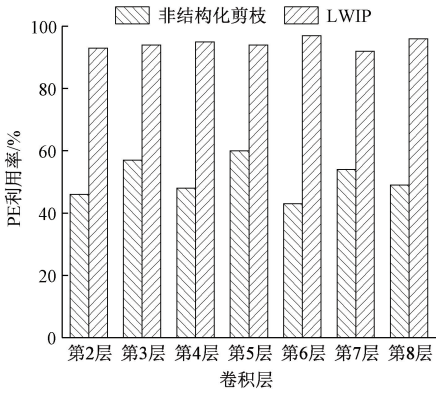

图 7　LWIP 算法的负载平衡能力

3.2　硬件性能与加速效果分析

本文使用 Verilog 硬件描述语言实现了提出的硬件加速器，在 Xilinx Vivado 2019.1 开发套件上完成了功能仿真和时序分析，最终将其部署在 Xilinx XC7Z100 FPGA 开发板上。使用 C 语言编写了软件驱动代码，并运行在 ARM 处理器上，以控制加速器执行 CNN 推理任务。输出并行度 N 设置为 16，即每个 PE 组中包含 16 个 PE；输入特征图并行度 M 设置为 32，即稀疏卷积单元中包含 32 个 PE 组。

为了验证本文所提出的加速器的性能优势，选择其他的 CNN 加速器研究进行比较，结果如表 3 所示。其中文献[4,6]是关于密集型 CNN 加速器的研究，文献[14,18,21]则是聚焦于稀疏 CNN 加速器的研究。为了更公平地与其他的稀疏 CNN 加速技术研究比较，本文选用与文献[14,18,21]中相同的 VGG16 网络作为比较对象。在比较的指标中，理论吞吐率表示加速器的理论算力，即数字信号处理器(digital signal processor，DSP)的数量与时钟频率乘积的二倍。但是由于文献[4]和本文优化了乘法器设计，文献[6]使用了快速卷积算法，因此实际吞吐率均高于理论吞吐率。同时，本文分别计算 DSP 和块随机存储器(block random access memory，BRAM)的效率来表示其平均吞吐率，以此来衡量不同加速器在单位数量硬件下的吞吐率。

表 3　与其他相关研究的性能比较

加速器	文献[4]	文献[6]	文献[14]	文献[21]	文献[18]	本文	本文
FPGA 型号	ZU5EV	PYNQ-Z2	ZCU102	ZCU102	ZCU102	XC7Z100	XC7Z100
CNN 类型	YOLOv3-Tiny	退化 YOLO	VGG16	VGG16	VGG16	VGG16	LWIP+VGG16
时钟频率/MHz	200	150	200	200	200	220	220
DSP 数量	334	115	1 144	1 564	1 061	1 024	1 024
BRAM 数量	92.5	67	912	840	502	464	464
理论吞吐率/(GOP/s)	132	34.5	457	625.6	424.4	450	450
实际吞吐率/(GOP/s)	144	51.89	309	534.2	—	706	—
DSP 效率/(GOP/s)	0.43	0.45	0.27	0.34	—	0.68	—
BRAM 效率/(GOP/s)	1.56	0.77	0.33	0.63	—	1.52	—
等效吞吐率/(GOP/s)	—	—	954	1 170.4	1 054	—	1 302
等效 DSP 效率/(GOP/s)	—	—	0.83	0.75	0.99	—	1.27
等效 BRAM 效率/(GOP/s)	—	—	1.05	1.39	2.10	—	2.81

注：GOP 为硬件加速器的吞吐量单位，表示加、减、乘、除的次数。

本文首先使用未经 LWIP 优化的 VGG16 网络与其他的研究进行性能对比，本文提出的加速器最高可工作在 220 MHz 的时钟频率下，理论吞吐率和实际吞吐率分别为 450 GOP/s 和 702 GOP/s。本文对乘法器进行优化使其同时计算两个乘法，因此实际吞吐率超过了理论值，而文献[14,18,21]提出的稀疏加速器的吞吐率均低于其理论值。本文的加速器的实际吞吐率远高于文献[4]和[6]，这是由于本文使用的硬件资源更多，提供了更大的计算并行度。在 DSP 效率上，本文的加速器分别是文献[4]和[6]的 1.58 倍和 1.51 倍；在 BRAM 效率上，本文的加速器比文献[4]略低，这是因为本文加速器的特征数据切片生成器模块额外消耗了 BRAM，使总体 BRAM 数量上升，从而导致总体效率下降。与文献[14]和[21]提出的稀疏加速器相比，本文加速器在 DSP 效率上比它们分别高 1.51 倍和 1.0 倍，在 BRAM 效率上分别高 3.6 倍和 1.41 倍。因此，使用基于权重复用和数据切片的稀疏计算方法解决了稀疏 CNN 加速器中的乘法器不能被高效利用的问题，从而本文的加速器在 DSP 和 BRAM 效率上优于其他的加速器。

此外，本文使用 LWIP 算法对 VGG16 网络进行优化并与其他研究作加速效果对比。由于不同研究对网络模型的优化方法是不同的，为了更公平地比较不同加速器的

加速效果,本文将稀疏加速器的吞吐率进行转换,即用推理帧率与网络的计算量相乘得到等效吞吐率进行比较。相比较于文献[4]和[6]提出的密集型 CNN 加速器,本文的加速器在等效 DSP 效率上分别是文献[4]和[6]的 2.95 倍和 2.82 倍;在等效 BRAM 效率上,分别是它们的 1.80 倍和 3.65 倍。这表明本文的稀疏 CNN 加速器在跳过零权重后减少了无效的冗余计算,使硬件的利用率得以提升,相比于密集型 CNN 加速器最高实现了 3.65 倍的加速比。文献[14]使用的 DSP 数量和本文相差不多,但由于文献[14]中切片查找表的实现消耗了大量的存储资源,因此 BRAM 的使用量大概是本文的 2 倍。在等效 DSP 和 BRAM 效率方面,本文的加速器比文献[14]分别高出 53% 和 167%。文献[21]的加速器所使用的 DSP 数量是本文的 1.5 倍,等效吞吐率是本文的 0.9 倍,等效 DSP 和 BRAM 效率分别是本文的 0.59 倍和 0.49 倍。文献[18]所使用的 DSP 和 BRAM 数量与本文最接近,但是在硬件效率上,本文的加速器比文献[18]分别高 28%和 34%。本文提出的 LWIP 算法优化了网络模型,使硬件中的负载分布相对平衡;同时在硬件加速器中对乘法器的优化使其吞吐率翻倍,新提出的数据排列方法解决了零填充的问题从而使乘法器的效率进一步提升,稀疏卷积单元的可配置特性能够灵活地支持任意稀疏度的卷积计算。相比于其他的稀疏 CNN 加速器研究,本文中来自软件和硬件的协同优化最终使加速效果得到 28～167%的提升。

4 结　　论

本文针对稀疏 CNN 的不规则权重分布而导致的硬件计算负载失衡的问题,首先提出了一种细粒度的模型剪枝算法,该算法在保证模型精度的同时最大化程度让模型权重的分布变得规则。其次,设计了一种具有高效 PE 结构和稀疏度可配置的稀疏卷积单元,可配置性使其灵活地适应任意稀疏度的卷积计算,同时 PE 中乘法器的优化使其吞吐率翻倍,新提出的数据排列方法解决了零填充的问题。实验结果表明,本文提出的剪枝算法有效地减少了 CNN 的计算复杂度且精度损失较小,解决了加速器中计算负载不平衡的问题;本文的硬件加速器拥有优异的性能指标,在吞吐率和硬件效率上均高于其他相关研究中的加速器。

参考文献

[1] 刘秋月, 刘雪峰, 孙绍华. 基于阴影增强和注意力机制的高光谱图像分类[J]. 电子测量技术, 2023, 46(8): 14-23.

[2] 强栋, 王占刚. 基于改进 YOLOv5 的复杂场景多目标检测[J]. 电子测量技术, 2022, 45(23): 82-90.

[3] 付孟新, 郭世伟, 王泽兴,等. 基于一维卷积神经网络的列车异响识别系统研究[J]. 电子测量技术, 2023, 46(14): 9-17.

[4] 缪丹丹, 张鹏, 张鑫宇,等. 基于 ZYNQ 平台的通用卷积加速器设计[J]. 国外电子测量技术, 2022, 41(11): 72-77.

[5] 吴宇航, 何军. 基于 FPGA 加速的行为识别算法研究[J]. 电子测量技术, 2022, 45(13): 25-32.

[6] 王习东, 王国鹏, 王保昌,等. 基于 FPGA 与退化 YOLO 的手机镜片缺陷检测系统[J]. 电子测量技术, 2022, 45(18): 10-17.

[7] 杜忠文, 李庚霖, 蒋菡,等. 基于次级缓存的 SDRAM 调度策略的研究[J]. 电子测量技术, 2023, 46(14): 37-42.

[8] HAN S, MAO H, DALLY W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. Fiber, 2015, 56(4): 3-7.

[9] 刘钊, 孙洁娣, 温江涛. 基于多层面压缩深度神经网络的轴承故障诊断[J]. 电子测量与仪器学报, 2022, 36(7): 189-198.

[10] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network[J]. Advances in Neural Information Processing Systems, 2015, 28: 1135-1143.

[11] ANWAR S, HWANG K, SUNG W. Structured pruning of deep convolutional neural networks[J]. ACM Journal on Emerging Technologies in Computing Systems(JETC), 2017, 13(3): 1-18.

[12] LIN M, JI R, WANG Y, et al. Hrank: Filter pruning using high-rank feature map[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020: 1529-1538.

[13] LIU B, WANG M, FOROOSH H, et al. Sparse convolutional neural networks[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 806-814.

[14] LU L, XIE J, HUANG R, et al. An efficient hardware accelerator for sparse convolutional neural networks on FPGAs[C]. 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines(FCCM), IEEE, 2019: 17-25.

[15] WEN J, MA Y, WANG Z. An efficient FPGA accelerator optimized for high throughput sparse CNN

inference[C]. 2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), IEEE, 2020: 165-168.

[16] YANG C, MENG Y, HUO K, et al. A sparse CNN accelerator for eliminating redundant computations in intra- and inter-convolutional/pooling layers[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems, 2022, 30(12): 1902-1915.

[17] LI N. A high energy-efficiency inference accelerator exploiting sparse CNNs[C]. Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence, 2020: 534-541.

[18] SUN W, LIU D, ZOU Z, et al. Sense: Model-hardware codesign for accelerating sparse CNNs on systolic arrays[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2023, 31(4): 470-483.

[19] ZHANG S, DU Z, ZHANG L, et al. Cambricon-X: An accelerator for sparse neural networks[C]. 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture(MICRO), Taipei, Taiwan, 2016, 1-12.

[20] SUI X, LV Q, BAI Y, et al. A hardware-friendly low-bit power-of-two quantization method for CNNs and its FPGA implementation[J]. Sensors, 2022, 22(17): 6618.

[21] QASAIMEH M, ZAMBRENO J. An efficient hardware architecture for sparse convolution using linear feedback shift registers[C]. 2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors(ASAP), NJ, USA, 2021, 250-257.

作者简介

李宁，硕士研究生，主要研究方向为神经网络硬件加速技术、FPGA 开发。

E-mail: 2017213670@mail.hfut.edu.cn

肖昊(通信作者)，博士，教授，主要研究方向为专用硬件加速器、人工智能与安全芯片等。

E-mail: xiaohao@hfut.edu.cn