DOI:10.19651/j.cnki.emt.2415770

基于低成本 FPGA 的深度卷积神经网络加速器设计*

杨 统 肖 昊
(合肥工业大学微电子学院 合肥 230601)

摘 要:现有的深度卷积神经网络在推理过程中产生大量的层间特征数据。为了在嵌入式系统中保持实时处理,需要大量的片上存储来缓存层间特征映射。本文提出了一种层间特征压缩技术,以显著降低片外存储器访问带宽。此外,本文针对 FPGA 中 BRAM 的特点提出了一种通用性的卷积计算方案,并从电路层面做出了优化,既减少了访存次数又提高了 DSP 的计算效率,从而大幅提高了计算速度。与 CPU 运行 MobileNetV2 相比,文章提出的深度卷积神经网络加速器在性能上提升了 6.3 倍;与同类型的 DCNN 加速器相比,文章提出的 DCNN 加速器在 DSP 性能效率上分别提升了 17%和 156%。

关键词:深度卷积神经网络;现场可编程门阵列;深度学习
中图分类号:TN46 **文献标识码**:A **国家标准学科分类代码**:510.4030

Design of deep convolutional neural network accelerator based on low-cost FPGA

Yang Tong Xiao Hao
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: Existing DCNN generate a large amount of inter-layer feature data during inference. To maintain real-time processing on embedded systems, a significant amount of on-chip storage is required to cache inter-layer feature maps. This paper proposes an inter-layer feature compression technique to significantly reduce off-chip memory access bandwidth. Additionally, a generic convolution computation scheme tailored for BRAM in FPGA is proposed, with optimizations made at the circuit level to reduce memory accesses and improve DSP computational efficiency, thereby greatly enhancing computation speed. Compared to running MobileNetV2 on a CPU, the proposed DCNN accelerator in this paper achieves a performance improvement of 6.3 times; compared to other DCNN accelerators of the same type, the proposed DCNN accelerator in this paper achieves DSP performance efficiency improvements of 17% and 156%, respectively.

Keywords: deep convolutional neural network; field programmable gate array; deep learning

0 引 言

深度卷积神经网络(deep convolutional neural network, DCNN)以其较高的准确率在计算机视觉和模式识别领域得到了广泛的应用[1]。然而,为了达到较高的预测精度,最新的 DCNN 已经变得更加复杂和多分支,基于中央处理器(central processing unit, CPU)的推理平台因为较低的并行度导致高延迟,而图形处理器(graphics processing unit, GPU)计算会产生较大功耗,于是现场可编程门阵列(field programmable gate array, FPGA)逐渐成为对 DCNN 加速的理想平台[2]。在推理过程中,DCNN 生成了超过数百兆字节的层间数据。如果层间数据存储不当,会严重影响硬件性能。当 DCNN 部署在资源有限的计算平台上时,这种影响会变得更加严重[3]。受片上存储器大小的限制,许多 DCNN 的层间特征映射不可避免地在片上存储器和片外存储器之间交换[4]。这不仅会导致数据交换造成较大的处理延迟,而且还会大大增加设备的功耗[5]。

近年来,研究人员逐渐开发出了很多方法对 DCNN 进行加速,文献[6]提出了一种用一个模块计算多层卷积,可以让数据循环计算而不使用额外的资源。然而将所有的输入特征图数据存储到片上,对参数量越来越大的 DCNN 模型部署于资源有限的 FPGA 显然不切实际。文献[7]提出

收稿日期:2024-04-05
* 基金项目:国家自然科学基金(61974039)项目资助

乒乓存储的硬件架构,解决了数据流中数据不连续的问题,极大的增加了处理的速度。文献[8]提出一种层间循环使标准卷积计算单元和深度卷积计算单元并行工作的方法,并且为了提高加速器第 1 层的计算效率,提出了一种专用的硬件架构,有效的提高了计算效率,但是成本开销较大。因此相关研究人员为提高资源的效率开始寻找新的解决方案。文献[9]提出行缓存的数据重排的方法从而避免了大量重复的输入特征值从片外加载到双倍速率同步动态随机存储器(double data rate synchronous dynamic random access memory, DDR SDRAM)(简称 DDR)上,然而这种方法并不能同时适用于深度卷积和标准卷积。文献[10]提出了一种在 FPGA 上实现(DCNN)全流水加速的框架。该框架将卷积层、反卷积层和全连接层,通过利用不同级别的并行性映射到一个统一的架构中。但是成本开销仍然较大,部分计算单元闲置的时间较长。文献[11]提出用 Karatsuba 算法取代了传统加速器中的数字信号处理(digital signal processing, DSP)计算,虽然降低了 DSP 资源的使用,但对计算的速度有一定影响。文献[12]提出了一种多 PE 的行级流水线策略,不同块的计算可以重叠,显著提高了加速器的并行性,文献[13]提出使用硬件实现 8×8 离散余弦变换(discrete cosine transformation, DCT)将存储的数据转换到频域来压缩层间特征映射,通过量化去除 DCT 后的高频分量,利用稀疏矩阵压缩进一步压缩层间特征映射,但是 DCT 变换需要用到乘法器并且计算较为复杂对资源有限的 FPGA 造成了负担。文献[14]中提出一种高度定制的流硬件架构,该架构通过利用不同级别的并行性、层融合和充分利用 DSP 进一步优化,但是并未对访存和电路层面进行优化。文献[15]提出一种将原始的计算简化成了并行度 NC 的向量点积的方法,让计算和存储空间更加灵活,但是同样并不适用于深度卷积并且并未针对 FPGA 上资源特点做出优化。

在本文中,针对以上问题,提出一种以数据重排为基础的资源分配方案,将输入特征值与卷积核的数据按照 FPGA 上随机存取内存块(block random access memory, BRAM)的存储特点进行了分块处理减少了存储资源的使用和 DSP 在读取 BRAM 中存储资源的访存次数。同时提出一种新的数据流的方案,使 PE 模块可计算标准卷积和深度卷积,同时充分利用 DSP 的功能,在电路层面进行资源上的优化。最后提出基于离散小波变换(discrete wavelet transform,DWT)的压缩方案用来解决 DDR 带宽问题。

1　软件及压缩算法设计

1.1　低比特量化

近几年来主流深度训练的模型以 32 位的浮点数比较常见,但是在 FPGA 上运行 32 位的浮点数往往需要很大的资源开销,需要对浮点数进行量化操作。一般来说,对于权重的量化,由于权重的数据分布是静态的,MinMax 是实现较为方便的量化方法,直接找出 min 和 max 线性映射即可,MinMax 其实就是将浮点数直接映射到 int8 的数据范围,MinMax 方法由浮点映射到定点的数学表达如下:

$$Q = \frac{R}{S} + Z \tag{1}$$

$$S = \frac{R_{\max} - R_{\min}}{Q_{\max} - Q_{\min}} \tag{2}$$

$$Z = Q_{\max} - \frac{R_{\max}}{S} \tag{3}$$

式中:R 为真实浮点值 fp32,Q 为量化后的定点值 int8,Q 属于[−128,127],Z 为 0 浮点值对应的量化定点值;S 为定点量化后可表示的最小刻度。这种量化方式,主要关注浮点范围的最大值和最小值,然后通过尺度 S 线性映射。MinMax 量化应用于网络权重这样静态分布的数据的时候,对于网络推理最后的精度损失影响不大,且量化操作开销更小,量化过程效率更高。

1.2　基于 DWT 的压缩算法

常用的图像数据压缩方法主要有哈夫曼编码、行程编码、算术编码、离散傅里叶变换(discrete fourier transform, DFT)、DCT 以及 DWT 等[16]。近些年图像压缩通过 DFT、DCT、DWT 通过将空间域上的图像信号转换到频域中,然后在将图像的频域分解各个子带并对各个子带进行分析以得到想要的图像信息。本文中所用的 DWT 变换的过程如图 1 所示,图像进行 DWT 变换后分为 4 个子带(b、h、v、c),其包含图像主要信息的左上角子带(b)能够再次不断的进行 DWT 变换从而将其连续分解成许多不同分辨率的信号,这意味着可以通过控制小波变换的次数来实现不同的压缩率目标。左上角子带(b)存储着图像的主要信息,而其余 3 个子带(h、v、c)存储着细节信息。传统的 DWT 压缩算法对于主要信息(b)采用无损压缩,也就是霍夫曼编码等其他编码算法,对于细节信息采用有损压缩的方式。霍夫曼编码是实现理论上最高压缩比的方法,但是它的编码和解码的实现将会引入一个查找表,这对于硬件来说引入了相当大的开销,尤其是对于低成本 FPGA 来说,在压缩单元使用大量的存储资源和计算资源对加速器整体的性能来说反而是得不偿失;并且传统的压缩方法并没有与具有即时性的层间特征值相匹配,并不适于直接运用于加速器设计。经过了多次 DWT 变换之后的主要信息相对较小,这部分的压缩对于整体的压缩率影响较小,所以本文中对细节信息进行量化,主要信息保持不变。本文中相对于之前的压缩方案压缩率较低,但是资源使用更少,且本文中的压缩方案具有即时性,与 DCNN 加速器中实时变化的层间特征值相匹配,更适于在低成本 FPGA 中实现。

2　硬件加速器设计

2.1　硬件架构

本文的硬件架构如图 2 所示,主要包括输入特征图和

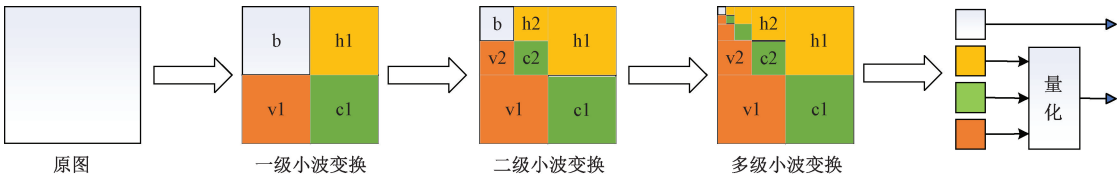

图 1　基于 DWT 变换的压缩方法

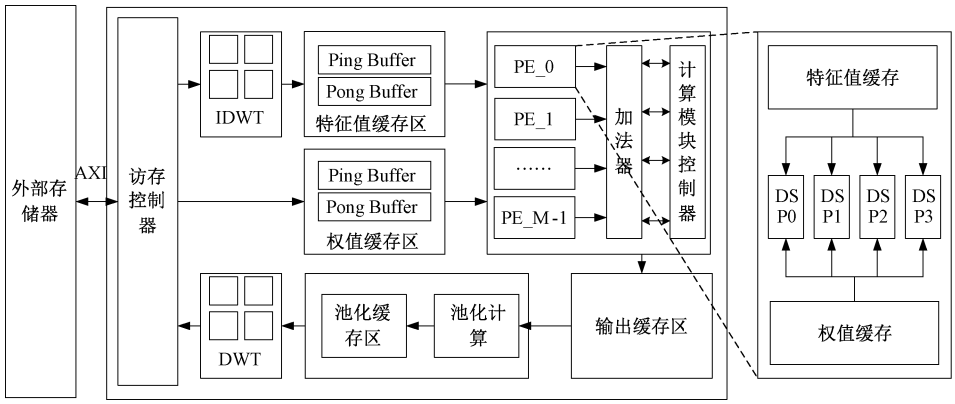

图 2　硬件架构图

权值的缓存区、PE 阵列、输出缓存区、池化区和压缩与解压缩区。权重和特征图数据存储在片外存储器中，片上存储器则负责对片外传来的数据进行缓存，一定数量的数据存储到片上后开始计算。

数据以 AXI 总线与 PL 端进行交互，当缓存区存取一定数量的特征图和权值后，PE 阵列开始计算，计算完成后的数据传输到输出缓存模块，输出缓存模块负责对输出特征图的激活，量化以及存储等工作。当这一层参数需要经过池化处理时的时候输出缓存区将数据送入到池化处理区，不需要池化则直接送入到 DWT 压缩模块，压缩完成之后的层间特征值再传回片外存储器中。

2.2　数据重排以及存储策略

在卷积神经网络加速器中缓存会被映 FPGA 特定的存储资源[17]，在 FPGA 上尤其是低成本 FPGA，存储资源有限，所以数据的缓存分块策略就显得重要[18]。本文对于输入特征图所用的分块策略如图 3 所示，对于大小为 $IW\times IH\times IC$ 的输入特征图，IW、IH 和 IC 分别表示输入特征图的宽、高和输入通道，一次提取 $K\times K\times 4M$ 个输入特征值（$K\leqslant IW$，$K\leqslant IH$），并将数据分成 M 个 $K\times K\times 4$ 的小块，将每个小块中的数据以图 3 所示的方式存入到 1 个 18 Kb 的 BRAM 中，本文中将不同输入通道上 4 个 8 bit 数据存入一个 18 Kb 大小的 BRAM 中的两个地址上，计算单元只需要一次读取，即可从缓存单元读出两个地址上的 4 个数据，减少了访存的次数。而对于标准卷积和深度卷积核的数据以不同的方式存储。对于大小为 $KW\times KH$ 的深度卷积核，以与输入特征值同样的方式将其分成 M 个 $KW\times KH\times 4$ 的小块，同样每个小块存入到 1 个 18 Kb 的 BRAM 中。而对于标准卷积则是把对应同一个输出通道与输入特征值相对应的 $4M$ 个数值存入到 1 个 18 Kb 的 BRAM。

2.3　处理单元 PE 设计

2019 年文献[7]用 DSP 的乘累加功能，让部分数据不在需要存储空间存储的同时减少了加法器的使用。在计算卷积核为 3×3 的深度卷积时，将 DSP 的输出结果与累加位相连接，在经过 9 个时钟周期后再输出结果。但是并没有利用 DSP 的乘累加设计相对应的电路对标准卷积进行计算，而是将点卷积和深度卷积的计算模块分开，浪费了计算单元的同时，计算类型也只能支持深度可分离卷积（先深度卷积再点卷积）。

本文中所设计的 PE 结构如图 4 所示，对于深度卷积和标准卷积使用了不同的映射方案，对于标准卷积由于一张特征图对应多组卷积核，所以利用特征图的复用性在计算普通卷积的时候只需要将 1 个输入特征缓存 BRAM 一次读取的 4 个数据分别复用到每一个 PE 计算单元，而对于深度卷积由于深度卷积中特征图与卷积核一一对应，所以每次需要将不同输入特征缓存 BRAM 中相同位置的数据读出送入到不同的 PE 单元。本文中单个 PE 单元的电路如图 5 所示，图中左侧 DSP 中的 A、B 为乘法位，分别对应着输入特征图和权值，C 为累加位，输出 P 有 3 条通路，连接着 3 个选择器，C1、C2、C3 均由卷积核类型和大小决定，在卷积核大小为 $KW\times KH$ 的标准卷积中输出位通过与 C 的相连，C1 控制数据经过 $KW\times KH\times M$ 次乘累加运算后将 4 个 DSP 生成的 4 个数据经过 C3 控制送入到一个由 3 个加法器组成的加法树单元中，计算完成后将 1 个数据输出到输出缓存单元。而在卷积核大小为 $KW\times KH$ 的深度卷积中，C1 控制数据经过 $KW\times KH$ 次乘累加运

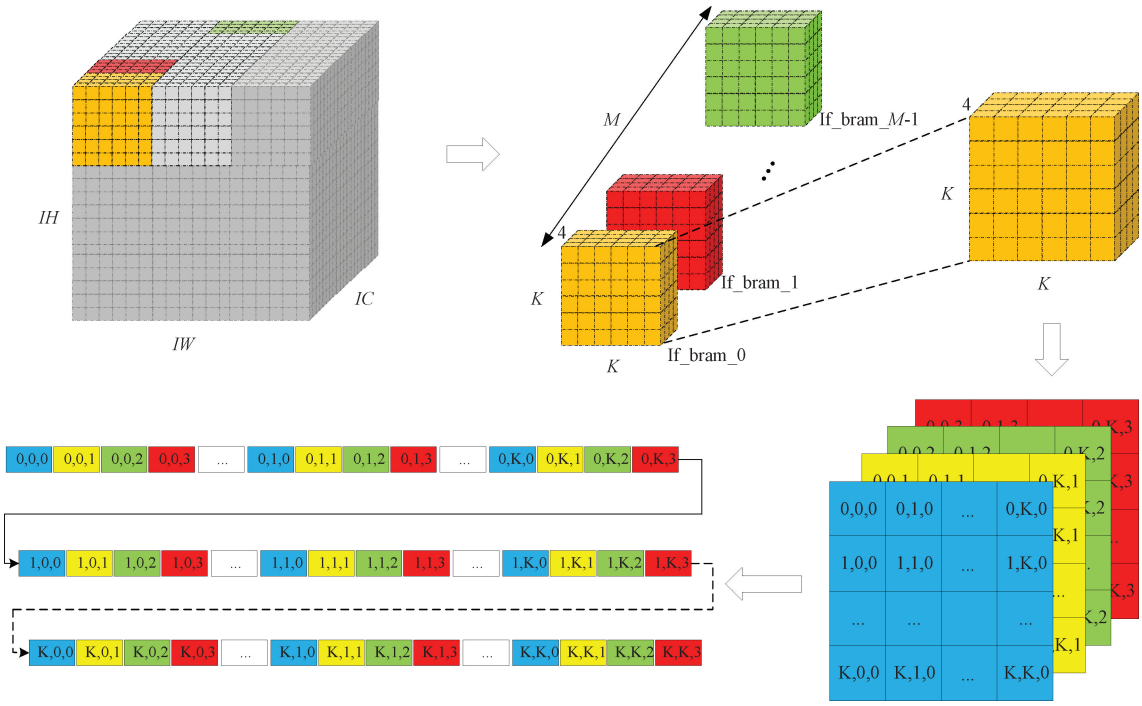

图 3　基于数据重排的分块策略

算后将 4 个数据通过 C2 控制将数据直接输出到输出缓存单元。传统的 PE 计算单元使用 DSP 与加法树相结合的电路进行卷积的乘累加计算,这种方法计算控制方便但是加法树的设置并不能同时适应于深度卷积核标准卷积,而本文中利用 DSP 的累加器通过控制电路同时适用于深度卷积和标准卷积,并且本文中的电路对于两种卷积的计算通路共用,相对于之前的通用型卷积计算电路减少了资源使用。总的来说,本文中所设计的 PE 单元可以支持不同类型和不同大小卷积核的卷积运算,并且将 DSP 的乘累加功能运用到了不同类型的卷积运算,减少了资源的使用和提高了资源的性能效率。

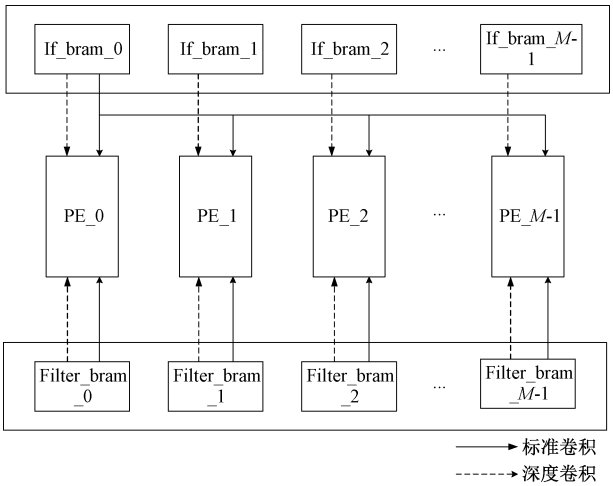

图 4　PE 单元结构图

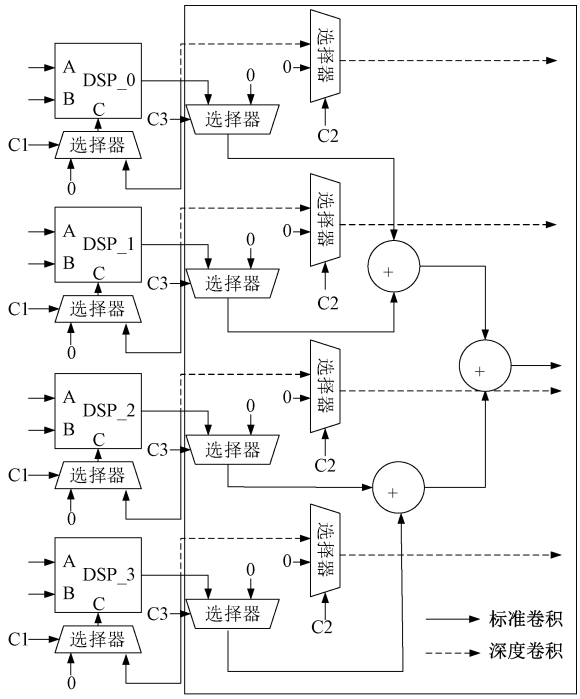

图 5　单个 PE 电路图

2.4　输出缓存单元设计

本设计中的输入特征图与权值缓存为了减少访存次数占用了过多的存储资源,因此为了节约存储资源与后续 DWT 压缩相衔接对输出缓存进行优化。乘累加之后的结果位宽较大,所以在传入到输出缓存之前对计算结果进行激活和量化。输入特征图的大小为 $K \times K \times 4\ M$,为了层

之间特征值传输，数据重排后数据流的一致性，输出特征图的大小保持和输入一致，但是由于各模块数据流并行输出缓存没有访存压力，于是本文将 $K\times K\times 4M$ 的块进一步进行分成 4 个区域 a、b、c、d，每个区域的大小为 $K/2\times K/2\times 4M$，设置 M 的被除数 N，每个小块用 N 个 18 Kb 的 BRAM 存储。通过可控的参数设置可以对资源分配进行优化。

2.5　DWT 压缩模块设计

DWT 变换的公式如下：

$$A_{ij}=\frac{a_{ij}+b_{ij}+c_{ij}+d_{ij}}{2} \tag{4}$$

$$B_{ij}=\frac{a_{ij}-b_{ij}+c_{ij}-d_{ij}}{2} \tag{5}$$

$$C_{ij}=\frac{a_{ij}+b_{ij}-c_{ij}-d_{ij}}{2} \tag{6}$$

$$D_{ij}=\frac{a_{ij}-b_{ij}-c_{ij}+d_{ij}}{2} \tag{7}$$

DWT 压缩模块中包含了 N 个如图 6 所示的电路，电路经过两级的 4 个的加法器，对于加法器的 4 个输出根据量化的程度进行移位操作然后再将结果缓存到相对应的 BRAM 上。本文中所得到的 A 模块仍由 N 个 BRAM 存储，B、C、D 模块则因为量化后一个地址可以存储更多数据。

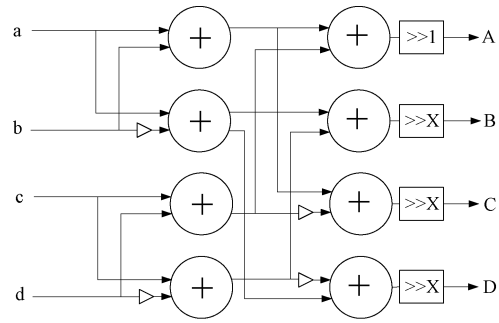

图 6　DWT 模块电路图

DWT 压缩模块的具体变换过程如图 7 所示，图中以一个 8×8 的为位宽为 8 bit 的图像为例，首先将 8×8 的图像分为 4 个 4×4 的小块，经过一次 DWT 变换后图像变为了 4 个 4×4 位宽为 9 bit 的小块 A、B、C、D，将 B、C、D 中的像素量化为 4 bit，量化后图像变为了一个 4×4 位宽为 9 bit 和 3 个 4×4 位宽为 4 bit 的 4 个小块，然后对 A 重复整个变换过程即可得到二级小波变化的压缩结果。

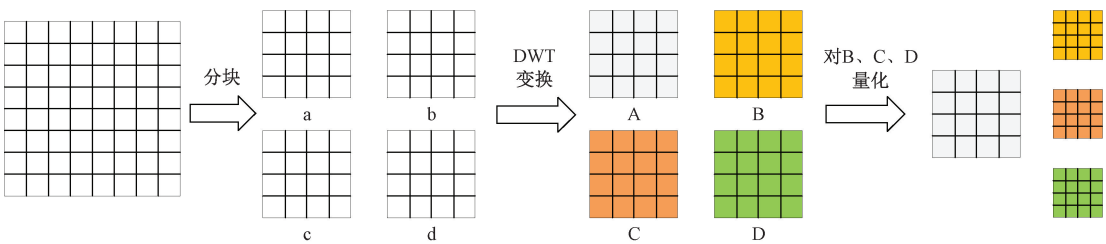

图 7　8×8 图像的 DWT 压缩

3　实验与评估

3.1　DWT 压缩算法评估

本文在 Pytorch 深度学习框架上实现了压缩算法，选用了 VGG16、MobileNetV1 和 MobileNetV2 3 种网络模型在 Classification 分类任务上对层间特征值压缩解压测试，由于在前十个融合层的尺寸远大于其他层并且方便与其他压缩方法做对比，本文中的压缩实验主要集中于前十层。由于 DWT 量化程度过高和小波变化次数过多时精度损失太大并会浪费资源影响计算速度，不适于低成本 FPGA，本文中仅量化为 4 bit 数据分别进行 1 次、2 次、3 次小波变化，并对实验结果的压缩比和精度损失与其他实验做出对比。不同压缩方法压缩比和精度损失的对比如表 1、2 所示。

$$\text{压缩比}=\frac{\text{压缩后的数据}}{\text{原始数据}} \tag{8}$$

与文献[19]本文的压缩比均不如，但是压缩算法逻辑较为复杂并且它使用一种专用硬件，将片内数据压缩到片外存储器，但是并没有结合 DCNN 加速器实时压缩，所以

表 1　压缩比对比　%

压缩方法	文献[13]	文献[19]	DWT-1	DWT-2	DWT-3
VGG16	30.63	34.36	65.62	54.68	51.36
MobileNetV1	61.02	N/A	65.62	54.68	51.36
MobileNetV2	71.05	40.81	65.62	54.68	51.36

表 2　精度损失对比　%

压缩方法	文献[11]	DWT-1	DWT-2	DWT-3
VGG16	0.45	0.23	0.45	0.61
MobileNetV1	0.44	0.25	0.48	0.59
MobileNetV2	0.49	0.28	0.47	0.62

本文主要与同样结合 DCNN 加速器做实时压缩的文献[13]做对比。相对于文献[13]，在精度损失不超过 0.5%的情况下，本文所提出的压缩方法在 VGG16 上压缩率较差，但是在 MobileNetV1 与 MobileNetV2 上优于文献[13]，并且本文的压缩方法压缩率更具有稳定性，可以通过公式得：

$$压缩率=\frac{m+n+(4^n-1)m_0}{m\times 4^n} \tag{9}$$

其中,m 为量化前的位宽,m_0 为量化之后的位宽,而 n 为 DWT 变换次数。

3.2　硬件性能与加速效果对比

本实验在集成开发环境 Pycharm 下通过调 Pytorch 中的库函数搭建了实验所需的 DCNN 模型。验证加速器性能时所采用的 FPGA 开发平台是基于 Xilinx Zynq-7020 扩展式处理平台的低成本开发板 Zedboard,DCNN 模型使用的是 MobileNetV2,并行度 M 设置为 48,也就是说卷积计算模块包含 48 个 PE,输出缓存的分块 N 设置为 6,只进行一次小波变化。DCNN 加速器的各个缓存模块在 Zedborad 上的 BRAM 资源消耗情况如表 3 所示。

表 3　资源损耗情况

资源类型	BRAM 个数	消耗 36 Kb BRAM 个数
输入缓存模块	96	48
卷积核缓存模块	96	48
输出图缓存模块	24	12
DWT 压缩模块	15	7.5
池化模块	12	6
IDWT 解压模块	15	7.5

从表 3 中可以看出在输入缓存模块与卷积核缓存模块占用了绝大部分的缓存 BRAM,而本文中设计的压缩模块只占用了 11%左右,压缩成本较低。

DCNN 加速器的整体资源消耗如表 4 所示。通过表 4 可知由于本文通过 DSP 的乘累加取代了一部分加法器资源且压缩算法逻辑不复杂所以本文中 LUT 资源消耗不多。

表 4　资源损耗情况

资源类型	资源总量	资源消耗	资源利用率
LUT	53 200	32 454	61.00
FF	106 400	57 382	53.93
DSP	220	192	87.27
BRAM	140	129	92.14

为验证本文提出的加速器性能,分别对比了 CPU 和 FPGA 上运行 MobileNetV2 的运行速率,在 CPU 上运行的速率为 14 fps,本文则为 88.3 fps,为在 CPU 上运行的 6.3 倍。与其他已有的基于 FPGA 的 DCNN 加速器进行对比的结果如表 5 所示,文献[8]中采用了流水线技术设计了相应的 DCNN 加速器,但是并没有考虑 BRAM 的访存次数,即同一个地址存放多个数据,导致 BRAM 使用过多。文献[15]中虽然 DSP 的性能效率很高,但是使用了大量的 BRAM 资源,并且量化程度过大,会导致模型精度太低,并不适合在低成本 FPGA 上实现。文献[10]上只对系统功耗做了优化,并未对 DSP 的效率做出优化,所以性能效率较差。文献[14]中同样没有考虑带宽问题和访存次数问题,虽然检测速度很快,但是资源损耗过大,整体的 DSP 性能效率不高。为了更公平的比较 DCNN 加速器的性能,本文使用 DSP 性能效率(检测速度与 DSP 个数的比值)这一指标来比较性能,本文所提出的加速器在性能效率上分别相对于文献[8]与文献[14]提升了 17%与 156%。由此可见本文所提出的方案在低成本 FPGA 上相对于 CPU 和其他方案都有显著优势,同时本文所提出的方案在 FPGA 上的资源利用率也因为可控的并行度设置而更高。

表 5　资源与性能对比

参数	文献[8]	文献[15]	文献[10]	文献[14]	本文
FPGA 型号	XCZU9EG	Arria 10 GX1150	XCZ7Z020	Intel GX1150	XC7Z020
网络模型	MobileNetV2	VGG16	MobileNetV2	MobileNetV2	MobileNetV2
数据位宽/Bit	8	4	16	8	8
DSP 消耗个数	2 070	147	199	1 518	192
BRAM 消耗个数	771	1 470	137	2 334	129
检测速度/FPS	810.3	112.1	16	280.4	88.3
DSP 性能效率	0.39	1.31	0.08	0.18	0.46

4　结　　论

本文设计了一种基于低成本 FPGA 的 DCNN 加速器,采用软硬件结合的方式将数据循环式的从 DDR 中读取和回流,基于数据重排重新设计了数据流与 PE 模块,使加速器可以利用 DSP 的乘累加计算深度卷积和标准卷积,减少了资源的使用,并提出了一种压缩效果较好的压缩算法 DWT 进行压缩以解决层间特征值传输时带宽不够的问题。实验结果表明,所提出的加速器在低成本 FPGA 上的适用性很强,在网络模型精度损失很小的情况能有效地提高深度卷积神经网络的计算速度,在 DSP 的性能效率上和 BRAM 资源的使用要优于其他相关研究。

参考文献

[1]　关忠榜, 杨颜博, 李敏超. 基于改进 Mask R-CNN 的

牛脸目标检测算法[J]. 电子测量技术，2023，46(24)：133-138.

[2] 王铮帅，邱联奎，李迎港. 复杂环境下的 YOLOv5s 烟火检测方法[J]. 电子测量技术，2023，46(24)：149-156.

[3] SANDLER M，HOWARD A，ZHU M，et al. MobileNetV2：Inverted residuals and linear bottlenecks[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition，2018：4510-4520.

[4] YIN S. A high energy efficient reconfigurable hybrid neuralnetwork processor for deep learning applications[J]. IEEE Solid-StateCircuits，2018，53(4)：968-982.

[5] 冉险生，李锐，贺帅. 基于改进 YOLOv5s 的道路障碍物检测算法[J].电子测量技术，2023，46(22)：177-185.

[6] SHEN Y，FERDMAN M，MILDE P. Maximizing CNN accelerator efficiency through resource partitioning[C]. 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture(ISCA)，2017：535-547.

[7] BAI L，ZHAO Y，HUANG X. A CNN accelerator on FPGA using depthwise separable convolution[J]. IEEE Transactions on Circuits and Systems II：Express Briefs，2018，65(10)：1415-1419.

[8] WU D，ZHANG Y，JIA X，et al. A high-performance CNN processor based on FPGA for MobileNets[C]. 2019 29th International Conference on Field Programmable Logic and Applications，2019：136-143.

[9] ZHANG Y，JIANG H，LI X，et al. Energy-efficient CNNs accelerator implementation on FPGA with optimized storage and dataflow[C]. 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications，Big Data & Cloud Computing，Sustainable Computing & Communications，Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)，2021：1209-1214.

[10] XIE X，ZHAO G，WEI W，et al. MobileNetV2 accelerator for power and speed balanced embedded applications[C]. 2022 IEEE 2nd International Conference on Data Science and Computer Application(ICDSCA)，2022：134-139.

[11] ARCHANA V S. An FPGA-based computation-efficient convolutional neural network accelerator[C]. 2022 IEEE International Power and Renewable Energy Conference(IPRECON)，2022：1-4.

[12] HUANG W. FPGA-based high-throughput CNN hardware accelerator with high computing resource utilization ratio[J]. IEEE Transactions on Neural Networks and Learning Systems，2022，33(8)：4069-4083.

[13] SHAO Z. Memory-efficient CNN accelerator based on interlayer feature map compression[J]. IEEE Transactions on Circuits and Systems I：Regular Papers，2022，69(2)：668-681.

[14] LIU S，FAN H，FERIANC M，et al. Toward full-stack acceleration of deep convolutional neural networks on FPGAs[J]. IEEE Transactions on Neural Networks and Learning Systems，2022，33(8)：3974-3987.

[15] BAI HAO-RAN. A flexible and low-resource CNN accelerator on FPGA for edge computing[C]. 2023 3rd International Conference on Neural Networks，Information and Communication Engineering(NNICE)，2023：646-650.

[16] 李晨，许雪，郭业才. 基于深度神经网络的单幅图像盲去噪算法[J]. 电子测量技术，2023，46(21)：183-192.

[17] 谢国波，郑晓锋，林志毅，等. 基于改进 YOLOv4 算法的高压塔鸟巢检测[J]. 电子测量技术，2022，45(18)：145-152.

[18] 钱磊，吴昊，乔晓强，等. 基于特征融合的调制识别增强与迁移演化[J]. 电子测量技术，2022，45(18)：153-160.

[19] XIONG F. STC：Significance-aware transform-based codec framework for external memory access reduction[C]. 2020 57th ACM/IEEE Design Automation Conference(DAC)，San Francisco，2020：1-6.

作者简介

杨统，硕士研究生，主要研究方向为神经网络加速。
E-mail：1051980394@qq.com

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