计算机科学
COMPUTER SCIENCE

面向边缘计算的轻量级网络硬件加速设计

余运俊, 张鹏飞, 龚汉城, 陈敏

引用本文

余运俊, 张鹏飞, 龚汉城, 陈敏. 面向边缘计算的轻量级网络硬件加速设计[J]. 计算机科学, 2023, 50(11A): 220800045-7.

YU Yunjun, ZHANG Pengfei, GONG Hancheng, CHEN Min. Lightweight Network Hardware Acceleration Design for Edge Computing [J]. Computer Science, 2023, 50(11A): 220800045-7.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220800045

面向边缘计算的轻量级网络硬件加速设计

余运俊[1]　张鹏飞[1]　龚汉城[2]　陈　敏[2]
1 南昌大学信息工程学院　南昌 330000
2 江西江投数字经济研究院　南昌 330000

摘　要　随着边缘设备数据的增多和神经网络的不断落地应用,边缘计算为以云计算为核心的大数据技术分担了压力。现场可编程门阵列(FPGA)因灵活的体系结构和低功耗,在边缘计算以及构建神经网络加速器中显示出优异的特性。但是,传统的基于传统卷积算法的 FPGA 解决方案往往受到片上计算单元数量的限制。使用 Zynq 作为硬件加速平台,对参数进行定点量化,利用数组分区提高流水线运行速度。采用 Winograd 快速卷积算法对传统的卷积进行改进,将卷积运算中的乘法运算转换为加法运算,降低了模型的计算复杂度,极大提高了所设计的加速器的计算性能。实验表明,XC7Z035 工作在 150MHz 时钟下获得了 43.5GOP/s 的性能,能效是 Xeon(R) Silver 4214R 的 129 倍,是双核 ARM 的 159 倍。所提方案在资源和功耗受限的情况下可以提供较高的性能,适用于网络边缘端对轻量级神经网络的落地应用。

关键词:边缘计算;硬件加速;轻量级卷积神经网络;Winograd;FPGA
中图法分类号　TP391

Lightweight Network Hardware Acceleration Design for Edge Computing

YU Yunjun[1],ZHANG Pengfei[1],GONG Hancheng[2] and CHEN Min[2]
1 School of Information Engineering,Nanchang University,Nanchang 330000,China
2 Jiangxi Jiangtou Digital Economy Research Institute,Nanchang 330000,China

Abstract　With the increase of edge device data and the continuous application of neural networks,the rise of edge computing has shared the pressure on big data technologies with cloud computing as the core. Field programmable gate arrays(FPGAs) have shown excellent properties in edge computing and building neural network accelerators due to their flexible architecture and low power consumption. But traditional FPGA solutions based on traditional convolution algorithms are often limited by the number of on-chip computing units. In this paper,Zynq is used as a hardware acceleration platform,to quantize parameters at a fixed point,and array partitioning is used to improve pipeline running speed. The Winograd fast convolution algorithm is used to improve the traditional convolution,and the multiplication operation in the convolution operation is converted into an addition operation,which reduces the computational complexity of the model. The computational performance of the designed accelerator is greatly improved. Experiments show that XC7Z035 can achieve 43.5GOP/s performance under 150MHz clock,and the energy efficiency is 129 times of Xeon(R) Silver 4214R and 159 timesof dual-core ARM. The proposedsolution is limited in resources and power consumption. It can provide high performance and is suitable for the landing application of lightweight neural networks at the edge of the network.

Keywords　Edge computing,Hardware acceleration,Lightweight convolutional neural networks,Winograd,FPGA

1　引言

卷积神经网络(Convolutional Neural Network,CNN)算法在图像处理、分类识别、目标追踪等领域得到广泛的应用[1]。随着深度学习的普及和 Caffe,Tensorflow,Torch 等深度学习框架的成熟,卷积神经网络模型的识别精度越来越高。常见的卷积神经网络主要由卷积层、池化层、全连接层组成。卷积层和池化层的主要功能是提取图像特征,全连接层通常用于分类。比较有名的有:LeNet-5[2]手写数字识别卷积神经网络,精度达 99%以上;AlexNet[3]模型和 VGG-16[4]模型突破了传统图像识别的精度;GooLeNet[5]和 ResNet[6]推动了卷积神经网络的应用。但是卷积神经网络参数越来越多,计算量逐渐增大,导致以云计算为核心的大数据处理技术难以处理庞大的边缘数据。边缘计算技术的提出,为云计算提供了有益补充。边缘计算是将应用、数据和服务从集中式网络节点推向网络边缘,在靠近设备和数据源头,融合网络存储应用核心能力的开放平台,就近提供智能服务[7]。在计算资源与内存带宽受限的情况下,边缘计算与卷积神经网络算法前向推理硬件加速的研究引起了越来越多的关注。目前,用于卷积神经网络加速的平台主要有 3 种:图形处理器(Graphic

基金项目:国家国际科技合作专项(2014DFG72240);江西省重点研发计划项目(20214BBG74006)
This work was supported by the National International Science and Technology Cooperation Project(2014DFG72240) and Key R&D Program in Jiangxi Province(20214BBG74006).
通信作者:余运俊(yuyunjun@ncu.edu.cn)

Processing Unit,GPU),它的软件可编程和多 CUDA 架构非常适合于卷积神经网络的加速,但巨大的功耗使得其很难集成到功耗有限的嵌入式平台[8];专用集成电路(Application Specific Integrated Circuit,ASIC),具有高性能、低功耗的特点,但其设计周期长,制造成本高[9];现场可编程门阵列(Field Programmable Gate Array,FPGA),具有低功耗、高灵活性等特点,已成为研究卷积神经网络硬件加速最受欢迎的平台。经过几十年的发展,FPGA 在数字信号处理、复杂的密集型计算领域得到了广泛应用[10]。同时大量的研究表明,基于 FPGA 的 CNN 推理加速器能够以灵活的数据精度、较低的功耗和较短的部署周期提供海量的计算资源[11]。设计高性能的基于 FPGA 的 CNN 加速器的关键挑战之一,是充分利用片上计算资源来加速卷积层中涉及的乘法累加运算,这两层运算占大多数 CNN 总运算量的 90%以上[12]。因此,神经网络加速器的优化重点是采取合适的卷积算法。目前卷积算法的实现主要分为非快速算法和快速算法。非快速算法包括空间卷积算法和通用矩阵乘法算法(General Matrix Multiplication,GEMM);快速算法包括 FFT(Fast Fourier Transform)算法和 Winograd 算法。虽然在 FPGA 的卷积实现中非快速算法被广泛应用,但是空间卷积算法和通用矩阵乘法都受到片上 DSP(Digital Signal Processing)数量的限制。为了进一步提高加速器的运算速度,人们开始更加关注快速算法[13]。相较于非快速算法,快速算法可以明显减少卷积操作中的乘法次数,提高加速比。当应用 FFT 和 Winograd 算法时,将输入的特征映射和滤波器转换到相应的域,进行逐元素矩阵乘法(EWMM)。例如,使用 6×6 输入块的 Winograd 算法可以将 3×3 滤波器的乘法次数缩减为传统卷积算法的 1/4,使用 8×8 输入块的 FFT 算法可以将 3×3 滤波器的乘法次数至少缩减为传统卷积算法的 1/3。文献[14]表明,Winograd 快速算法和经典 FFT 算法可以显著降低算法复杂度。当输入和过滤器大小相同时,基于 FFT 的卷积是最优的,而基于 Winograd 的卷积最适用于过滤器较小且 stride 为 1 的情况。大多数流行的 CNN 模型在大多数层中使用较小的过滤器尺寸(5×5 或更小)进行卷积[3-4,6]。例如,MobileNet 和 YOLO 只使用 3×3filters[4,15],Resnet 和 Googlenet 广泛使用 3×3 和 5×5filters。本文基于卷积神经网络结构,改进了一种基于 Winograd 算法的面向边缘计算的卷积神经网络加速模块,通过引入双缓冲机制,对数据读写进行数组分区优化,实现加速模块各个部分的流水线运行;对权重参数做动态定点量化处理,兼顾性能与资源消耗;引入 Winograd 快速算法,降低计算复杂度,在性能效率与存储资源消耗上取得较好的加速效果,处理网络边缘数据并进行卷积神经网络硬件加速。

2 卷积神经网络结构

本文提出的轻量级神经网络,即参数量小、计算复杂度低、对计算资源和功耗要求不高的网络。典型的轻量级网络有 LeNet-5,Xception 和 MobileNet 系列等。卷积神经网络基本上已经成为用于各种计算机视觉任务的主导人工智能(Artificial Intelligence,AI)方法,例如图像分类[5]、人脸识别[16]、语义分割[17]和对象检测[18]。Le-Cun 提出了一个现代卷积神经网络 LeNet-5,如图 1 所示。

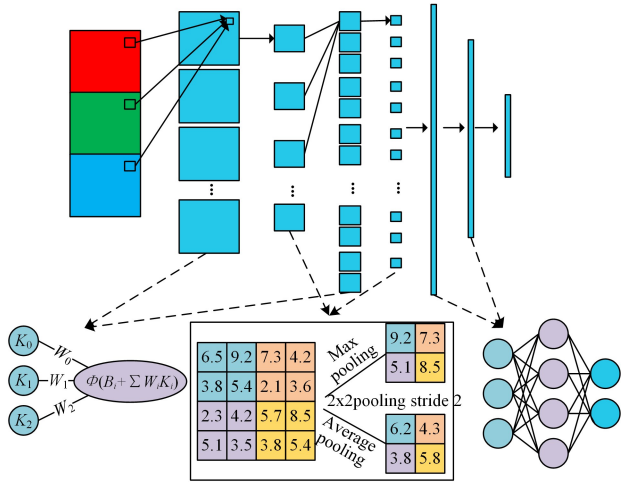

图 1 LeNet_5 神经网络结构图

Fig. 1 LeNet_5 neural network structure diagram

这个神经网络是一个 7 层的神经网络[19],与同大多数神经网络一样,如 AlexNet,GoogLeNet,MobileNet 和 ResNet。LeNet-5 的基本结构也是由输入层、卷积层、子采样层(池化层)、全连接层、输出层构成。下面针对这几个常用层对神经网络结构进行介绍。卷积层是实现神经网络功能关键的层结构之一,它通过卷积核对输入层的局部图片数据进行卷积操作,识别并提取局部特征,在卷积滤波器之后使用激活函数解决非线性问题,形成特征映射通道。卷积层在 CNN 中发挥着很重要的作用,一方面是权重共享,同一个映射通道内的神经元具有相同的参数,这极大地减少了 CNN 的参数总量。在同一个特征图的不同空间位置,可能具有一些相同的特征,如边缘、点、角度等。权值分配使 CNN 对位置和移动的敏感度降低。另一方面,每个卷积运算的目标是一小块输入,因此提取的特征保持有助于识别模式的输入的固有拓扑[20]。卷积操作如式(1)所示:

$$Y_j=\varphi(B_j+\sum_{i=0}^{n}W_{ij}*K_j) \tag{1}$$

其中,Y_j 代表的是卷积操作的输出结果,B_j 代表的是偏差,W_{ij} 是卷积核的内部参数,K_j 是卷积层的输入,$\varphi(\cdot)$ 是激活函数。激活函数之后一般是池化层,池化层的操作是降采样,目的是进行特征提取,总结相邻池化单元的输出[3]。常用的方法有最大值池化和平均池化两种,以 2×2 的池化窗口为例,最大值池化取窗口中的最大值,平均池化计算平均值。经过 2×2 的池化操作后,输出特征图宽和高各为原来的一半,不但提取了特征,还减少了参数量和计算量。

引入全连接层的目的是学习卷积层或最大池化层输出的局部非线性组合[21],减少特征位置不同对分类带来的影响。全连接层的每一个结点都与上一层的所有结点相连,用来把前边提取到的特征综合起来。由于其全相连,因此全连接层的参数也是最多的,参数的存储需要占用很大的内存。全连接层从计算特点上可以看作是一种特殊的卷积操作。输出层用来给出最后的神经网络预测的最终结果,所以输出层的输出神经单元一般是固定的。CNN 精度的显著提高,带来了巨大的计算复杂度,因为它需要对特征映射内的所有范围进行综合估计。面对如此巨大的资源和计算压力,GPU,FPGA 和 ASIC 等硬件加速器被用来加速 CNN。目前边缘计算设备大多采用混合 CPU-GPU-DLA 的异构架构,相比于单 CPU 或

者单 GPU 的架构，降低了功耗，提高了算力，但考虑到功能的迭代更新与应用场景的变换，FPGA 在可重构编程方面要优于混合 CPU-GPU-DLA 的异构架构。

3 神经网络硬件加速

针对加速器结构的设计，一方面是对软件层面即卷积网络的优化，大多数 CNN 模型的主要计算时间都花在卷积层上，而大部分参数来自全连接层，如图 2 所示。

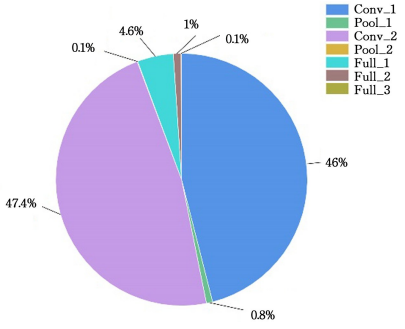

图 2 LeNet_5 各层运算时间占比图

Fig. 2 Proportion of computing time for each layer of LeNet_5

卷积层的运算量占据整个神经网络运算量的 93%，而卷积运算是频繁的乘累加操作。由于资源和带宽的限制，在 FPGA 上实现 CNN 不是一种有效的做法[22]。因此从优化参与运算的数据入手，本文提出对 CNN 参数动态定点量化的 FPGA 实现。

在 FPGA 上，乘法的计算会消耗更多的资源和时钟，往往是很慢的。一般的乘法需要借助 FPGA 中的 DSP 来计算，DSP 的大小有 18 bits×25 bits 的乘法，如果两个浮点数较大，则需要更多的乘法。采用 Winograd 快速卷积算法，降低计算复杂度，既可以节省 FPGA 的资源，也能取得不错的性能表现。另一方面，是对硬件层面的优化。CNN 具有高度层次化的多特征结构映射，其结构显示出高度的并行化。FPGA 使用可编程门阵列构建定制化处理单元，提供了低功耗和高度并行化的工作流程[23]，非常适合探索更高效的并行化神经网络加速电路。针对这一点，该文提出构建并行化 CNN 加速电路的实现，在有限的资源和功耗的情况下，探索 HLS 的最大优化策略。

3.1 FPGA 加速器结构

加速器采用 ARM＋FPGA 的异构架构，其主要由 PS 和 PL 两部分组成。系统采用共享内存方案，实现了 CPU 与 FPGA 之间的数据和指令通信。系统的整体架构如图 3 所示。

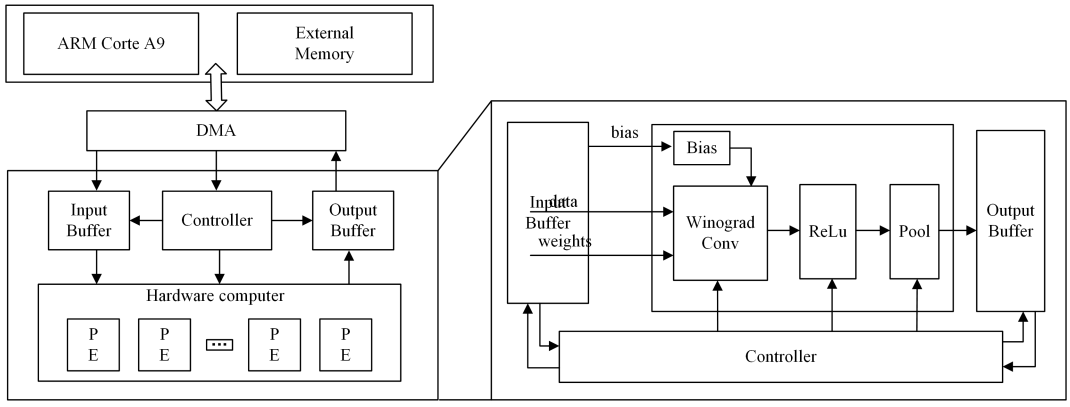

图 3 轻量级神经网络加速器架构

Fig. 3 Lightweight neural network accelerator architecture

PS 内部的 Cortex A9 处理器的任务是将网络参数的特征映射输入到 DDR 缓冲器中，并通过 AXI4_Lite 总线 Master 端访问 FPGA 加速器的状态寄存器并发送控制信号。PL 部分的 DMA 通过 AXI_Stream 总线访问 DDR，获取特征图及其卷积滤波器参数，并将这些数据缓存在片上 RAM(Input Buffer)中，Winograd Conv 模块负责对数据进行 Winograd 变换。为了后续 ReLu 和 Pool 的顺利进行，卷积结果经过反变换后向下传递。池化后的结果输出到 Output Buffer 单元，方便进行下一次运算。

本文提出的轻量级神经网络加速器是由 Vivado HLS 2018.3 生成的。硬件描述语言可以将 C 语言程序转换成现场可编程门阵列所需的 RTL 电路。硬件架构在 Vivado 2018.3 中构建。加速平台是在 Xilinx ZYNQ 板上开发的，主芯片为 XC7Z035-ffg676-2。

本文着眼于轻量级卷积神经网络的落地应用，在资源和功耗受限的情况下实现神经网络的推理过程。主要创新有以下几点：

(1)面向边缘计算，引入 Winograd 算法的同时，利用双缓冲机制和资源复用等技术，优化卷积过程中的数据缓存和数据传输速度；

(2)分析不同卷积层权重和偏置的数据范围，合理使用 HLS 工具提供的优化指令实现动态定点量化并显著提高加速器性能，提供准确推理结果的同时兼顾性能与资源消耗。

3.2 参数定点化

降低数据精度，对 NN 的参数进行量化，是提高 NN 加速器计算效率的有效方式[24]。基于 C 的原生数据类型以 8 位为边界(8 位、16 位、32 位、64 位)，虽然有规定好的数据类型和数据位宽，变量可以方便地进行数值传递和数据操作，降低代码的出错率，但是对硬件而言，这样的数据过于死板，浪费存储空间的同时也会降低硬件的运行效率。以一个实际位宽为 18＊18 的乘法器为例子，若在 HLS 中声明的数据类型是 32 bit 的浮点数，那么对应的输出就要声明为 64 bit。HLS 的综合结果显示，一个 32 bit 浮点数的乘法器占用的资源是定点数的 4 倍。所以在 FPGA 中用 32 bit 来描述 18 bit 是相当浪费资源的。使用浮点数，与任意精度定点数的效果对比如表 1 所列。

如表 1 所列，通过选择任意精度定点数，可以凭借不多的资源获得近乎相同的精度，同时还可以工作在更小的时钟周期下，改善数据吞吐率与运算速度。基于硬件有限的存储资源与逻辑资源，进行大规模的浮点运算是不现实的，因此选择一种合适的定点化方案，以较少的资源实现高效运算，同时降低数据精度的损失，就显得极为重要。位宽、溢出、量化模式的具体描述信息都可以使用 HLS 提供的任意精度定点运算 ap_fixed〈W，I，Q，O〉进行设置。任意精度定点数据类型的主要优势在于有助于 HLS 综合出更高质量的硬件。HLS 提供了多种量化模式与溢出模式。如量化模式 AP_RND，将值四舍五入到特定 ap_[u] 固定类型的最接近的可表示值；溢出模式 AP_SAT 使值饱和，从而在溢出时达到最大值，在负溢出的情况下为负最大值。以 LeNet-5 模型为例，统计训练好的网络权重。因为卷积神经网络各层权重之间有差异，所以本设计以层为单位进行数据的定点量化。确定 Lenet-5 网络的第一层权重的最小值和最大值分别为 −0.365 235 6 和 0.163 054 29，采用 8bit 的量化方式。结合图 4 所示的资源消耗对比情况，第一层卷积选择 8bit 量化方式可以最大限度地减少精度损失带来的推理准确率影响。

表 1 浮点数与任意精度定点数的对比

Table 1 Comparison of floating-point and arbitrary-precision fixed-point

	timing/ns	latency/ns	DSP48E	FF	LUT
float	8.43	2	4	3	1
fixed	5.09	0	1	0	0

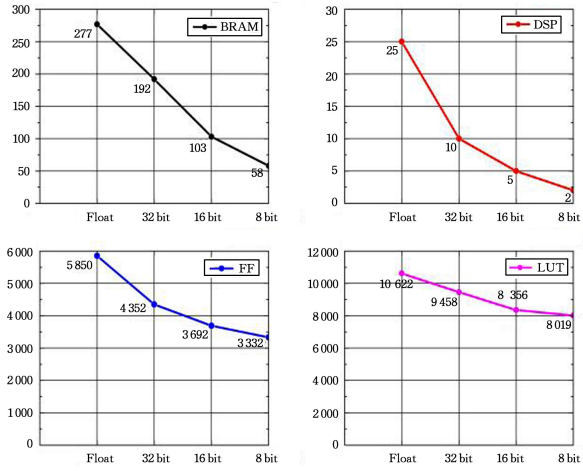

图 4 不同精度数据消耗资源对比

Fig. 4 Comparison of resource consumption with different precision data

3.3 数组分区改善流水线

在实际综合的过程中，某些 Loop 无法达到预先指定的初始时间间隔(II)，从而影响加速器的计算性能。上述现象往往是由数组导致的，数组作为最多只含有 2 各数据端口的 RAM 来实现。内存端口受限，无法同时读取同一个 RAM 块上的多个数据，这会限制神经网络这种读写(加载/存储)密集型算法的吞吐量。针对这种现象，本文用 HLS 提供的 ARRAY_PARTITION 数组分区指令对目标数组采取 Complete 方法分区，如图 5 所示。通过将数组(单一块 RAM 资源)拆分为多个更小的数组(多个 RAM 块)增加端口数量，有效改善带宽。

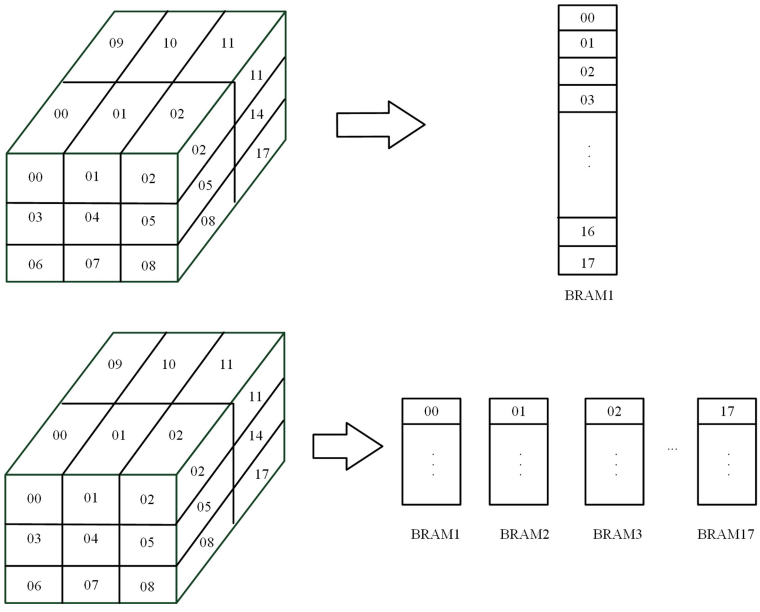

图 5 数组分区优化示意图

Fig. 5 Array partition optimization diagram

3.4 Winograd 快速卷积算法

Winograd 卷积是一种新的 CNN 快速算法，是一种基于中国余数定理的快速卷积算法。Winograd 算法通过输入图和卷积核上的一系列变换，用加法计算来代替乘法运算次数。以一维 Winograd 算法为例，对于一维的卷积过程 $F(m,r)$(其中，m 为输出矩阵尺寸，r 为滤波器尺寸)，将输入特征图和卷积核转换到 Winograd 域进行计算，降低乘法复杂度。对于 $F(2,3)$，假设输入数据为 $x=(x_0,x_1,x_2,x_3)$，卷积核为 $w=(w_0,w_1,w_2)$，输出数据为 $y=(y_0,y_1)$，则具体过程如下：

$$F(2,3)=\begin{bmatrix}x_0 & x_1 & x_2\\ x_1 & x_2 & x_3\end{bmatrix}\begin{bmatrix}w_0\\ w_1\\ w_2\end{bmatrix} \tag{2}$$

$$F(2,3)=\begin{bmatrix}m_1+m_2+m_3\\ m_2-m_3-m_4\end{bmatrix}=\begin{bmatrix}y_0\\ y_1\end{bmatrix}$$

其中，m 的值如式(3)所示：

$$\begin{cases} m_1=(x_0-x_2)y_0 \\ m_2=(x_1+x_2)\dfrac{(y_0+y_1+y_2)}{2} \\ m_3=(x_2-x_1)\dfrac{(y_0-y_1+y_2)}{2} \\ m_4=(x_1-x_3)y_2 \end{cases} \tag{3}$$

通过整理式(2)可得：

$$Y=A^{\mathrm{T}}[(GW)\odot(B^{\mathrm{T}}X)] \tag{4}$$

对于二维卷积过程 $F(m\times m,r\times r)$，其中 $r\times r$ 代表的是卷积核矩阵的大小，$m\times m$ 代表的是输出矩阵的大小，式(4)可以推广到二维卷积：

$$Y=A^{\mathrm{T}}[(GWG^{\mathrm{T}})\odot(B^{\mathrm{T}}XB)]A \tag{5}$$

式(5)即为 Winograd 二维卷积计算公式。其中 X 为输入特征图，W 为卷积核，$\odot$ 为对应位置相乘运算，A^{T}，B^{T}，G 分别是输出转换矩阵、输入转换矩阵和卷积核转换矩阵。参数矩阵如下：

$$\begin{cases} A^{\mathrm{T}}=\begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix} \\ B^{\mathrm{T}}=\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \\ G=\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{-1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} \end{cases} \tag{6}$$

将最小 1D 算法 $F(m,r)$ 嵌套于自身，得到最小 2D 算法 $F(m\times m,r\times r)$。Winograd 算法的计算过程如图 6 所示。

对于二维卷积，由于式(6)中的变换常数矩阵的值都为 1，0，−1，可简单地通过符号位取反和取零代替乘法。Winograd 算法的乘法次数如式(7)所示：

$$\begin{aligned} \mu(F(m\times m),(r\times r))&=\mu(F(m,r))\mu(F(m,r)) \\ &=(m+r+1)(m+r-1) \end{aligned} \tag{7}$$

相对于传统的滑动卷积所需的 $m\times m\times r\times r$ 次乘法，Winograd 算法所需的乘法次数下降到了 $(m+r-1)\times(m+r-1)$ 次。对于 $F(2\times 2,3\times 3)$，Winograd 算法只需要 16 次乘法，而传统的卷积算法则需要 36 次乘法，传统卷积算法的乘法次数是 Winograd 算法的 2.25 倍。Winograd 的伪代码如算法 1 所示。

算法 1 Winograd 快速卷积伪代码

```
输入：In[M][H][W]，K[ki][ko]
输出：Out[N][R][C]
1. for row=0；row<H；row+=1 do
2.    for col=0；col<H；col+=1 do
3.       for ki=0；ki<M；ki+=1 do
4.          for ko=0；ko<M；ko+=1 do
5.          Winograd(In，K，Out)
6.       end for
7.       end for
8.    end for
9. end for
10. Winograd(In，K，out){
11.          U=BᵀInB；
12.          V=GKGᵀ；
13. Out=Aᵀ[UV]A；}
```

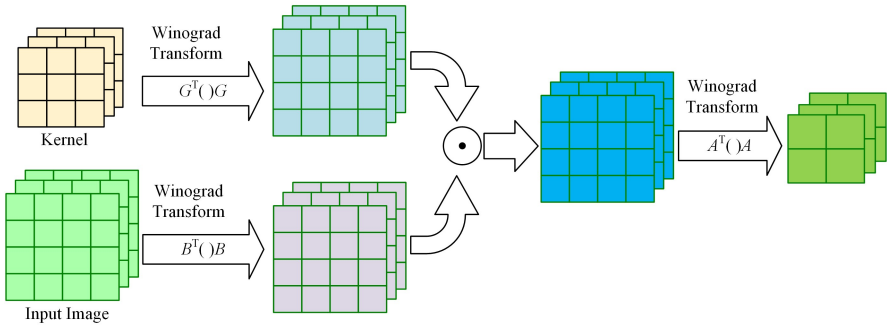

图 6 Winograd 的计算过程

Fig. 6 Calculation process of Winograd

4 实验与结果分析

4.1 实验环境

本文所测试的卷积神经网络是 LeNet-5，采用 Winogard 快速卷积算法和流水线并行优化的基本设计思想，以 Xilinx 公司的 ZynqZ7035 芯片为设计平台，使用 HLS 高层次开发工具搭建加速器电路。ZYNQ 芯片的 PL 端工作在 150MHz 的频率下。CPU 平台使用的是 DELL 塔式服务器 Intel(R) Xeon(R) Silver 4214R CPU @ 2.40 GHz，64 GB 内存。ZYNQ 平台的双核 ARM Cortex-A9 工作的时钟频率是 667 MHz。使用 Vivado HLS v2018.3 对加速器进行综合优化与布局布线。

4.2 资源耗费评估

经过综合优化后，资源消耗情况如表 2 所列。卷积运算是乘法密集型计算，主要是将消耗 DSP 资源作为评估资源利用率的主要因素。BRAM 资源主要用于实现对计算数据的缓存，例如 AXI 总线接口数据的缓存，以及图片输入数据的缓存等。

表 2 FPGA 资源耗费

Table 2 FPGA resource consumption

Resource	Avaliable	Utilization	Utilization/%
DSPs	900	649	72.11
BRAMs	500	81	16.2
LUTs	172×10^3	70.7×10^3	41.14
FFs	344×10^3	54.8×10^3	15.94

4.3 性能对比

为了进一步评估加速电路的计算性能，将本研究与近年的 CNN 加速器进行对比。

如表 3 所列，由于各个文献中使用的评估板和运算资源都有区别，数据精度和时钟频率也有差异，因此本文使用计算密度（单个 DSP 下的有效算力）来评估性能。文献[25]数据位宽大，加速器的效率受到带宽影响，整体性能较低。文献[26]设计了一个能适应 3×3 和 7×7 的卷积核，但对卷积层和全连接层衔接部分的设计不够精简，限制了计算速度的提升，降低了计算资源的利用率，所以性能仅 18.8GOPS。与其他工作对比，本文提出的方案在单个 DSP 上的计算密度和整体的计算性能均优于其他方案。综上所述，本文采用的方案极大地减少了卷积运算中的乘法运算数量，提高了加速器的计算性能。

表 3 相关工作比较

Table 3 Comparison of related works

	文献[25]	文献[26]	文献[27]	本文方案
Device	SX240t	ZC706	XC7Z045	XC7Z035
Frequency/MHz	120	200	150	150
Precision	32 bit fixed	32 bit fixed	16 bit fixed	16 bit fixed
Performance/GOPS	16.0	18.8	23.18	43.5
GOPS/DSP	0.015	0.023	0.026	0.067

如表 4 所列，与 CPU 平台相比，ZynqZ7035 的能效是 Xeon(R) Silver 4214R 的 129 倍，是 ARM 能效的 159 倍左右。通过 AIDA64 专业软件测得 Xeon(R) Silver 4214R 的峰值功是 57W 左右。48 核的处理单元，计算性能强，计算时延较短，但是功耗较高，导致能效低下。ARM 刚好相反，功耗较低，也导致了计算性能的不理想。本设计使用 ARM+FPGA 的异构加速器结构，降低了功耗的同时，有效缩短了卷积神经网络的计算时间。

表 4 与 CPU 的功耗和性能对比

Table 4 Comparison with CPU power consumption and performance

	Xeon(R)Silver 4214R	ARM_A9	XC7Z035
frequency/MHz	2400.00	666.67	150.00
Power/W	57.00	1.80	4.56
PerformanceGOP/s	4.20	0.11	43.5

结束语 轻量级卷积神经网络的加速器设计主要受边缘端计算资源与功耗的限制。本文以 LeNet-5 卷积神经网络算法为例，设计并实现了一种面向边缘计算的轻量级神经网络加速器架构。本设计结合 Winograd 快速卷积算法以及参数定点量化等方式，在减小计算量的同时降低了卷积神经网络前向计算对资源的消耗，提高了 FPGA 片上单个 DSP 的算力。实验表明，相同计算量情况下，ZynqZ7035 上获得了 43.5 GOP/s 的性能，能效是 Xeon(R) Silver 4214R 的 129 倍，是双核 ARM 的 159 倍。与近年来相关领域文献对比，本文提出的方案在资源和功耗受限的情况下可以提供更高的性能。同时，该加速器具有较高的应用推广价值面，为网络边缘端计算资源不足和轻量级神经网络的落地应用提供了一种解决方案。

该方案主要是着眼于卷积的优化，提高 DSP 的资源利用率。但本方案面向边缘计算，在资源和功耗都受限的情况下，对 LUT 资源利用不充分。下一步考虑从细粒度的角度出发，结合卷积神经网络各层算法的差异性，优化程序流水线结构，充分利用片上资源的同时提高吞吐率和计算性能，为边缘计算提供更优解决方案。

参 考 文 献

[1] GIRSHICK R, DONAHUE J, DARRELL T et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[C]// 2014 IEEE Conference on Computer Vision and Pattern Recognition 2014: 580-587.

[2] LECUN Y, BOTTOU L. Gradient-based learning applied to document recognition[C]// Proceedings of the IEEE. 1998: 2278-2324.

[3] KRIZHEVSKY A, SUTSKEVER I, HINTON G. ImageNet Classification with Deep Convolutional Neural Networks[C]// NIPS 2012. 2012.

[4] SIMONYAN K, ZISSERMAN A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. arXiv: 1409.1506, 2014.

[5] SZEGEDY C, LIU W, JIA Y et al. Going Deeper with Convolutions[C]// CVPR. 2015: 1-9.

[6] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition(CVPR). 2016: 770-778.

[7] LU Z, CHEN Y, LI T, et al. Convolutional Neural Network Construction Method for Embedded FPGAs Oriented Edge Computing[J]. Compute Research Develope, 2018, 55: 12.

[8] COATES A, HUVAL B, WANG T, et al. Deep learning with COTS HPC systems[C]// Proceedings of the 30th International Conference on International Conference on Machine Learning. JMLR. org, 2013: 1337-1345.

[9] JOUPPI N P, YOUNG C, PATIL N, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit[C]// the 44th Annual International Symposium, 2017: 1-12.

[10] AMIRI M, SIDDIQUI F M, KELLY C, et al. FPGA-Based Soft-Core Processors for Image Processing Applications[J]. Journal of Signal Processing Systems, 2017, 87: 139-156.

[11] SZE V, CHEN Y H, YANG T J, et al. Efficient Processing of Deep Neural Networks: A Tutorial and Survey[C]// Proceedings of the IEEE. 2017.

[12] YANG T J, CHEN Y H, SZE V. Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning [C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR). 2017.

[13] LU L Q, ZHENG S Z, XIAO Q C, et al. Accelerating convolutional neural networks on FPGAs[J]. Science China Information Sciences, 2019, 49: 277-294.

[14] LIANG Y, LU L, XIAO Q, et al. Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39: 857-870.

[15] REDMON J, DIVVALA S, GIRSHICK R, et al. You Only Look Once: Unified, Real-Time Object Detection[C]// CVPR. IEEE, 2016: 779-788.

[16] TAIGMAN Y, YANG M, RANZATO M, et al. DeepFace: Clos-

ing the Gap to Human-Level Performance in Face Verification [C]// 2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014:1701-1708.

[17] LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation[C]// 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015:3431-3440.

[18] REN S, HE K, GIRSHICK R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39:1137-1149.

[19] HAN S, POOL J, TRAN J, et al. Learning both Weights and Connections for Efficient Neural Networks[J]. Advances in Neural Information Processing Systems, 2015, 28:1135-1143.

[20] ZHANG Q, ZHANG M, CHEN T, et al. Recent Advances in Convolutional Neural Network Acceleration[J]. Neurocomputing, 2019, 323:37-51.

[21] YOUNG S, WANG Z, TAUBMAN D, et al. Transform Quantization for CNN Compression[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021, 44:5700-5714.

[22] CHENJ, LIUL, LIUY, etal. ALearningFrameworkfor〈italic〉n〈/italic〉-Bit Quantized Neural Networks Toward FPGAs[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 32:1067-1081.

[23] AYACHI R, SAID Y, BEN ABDELALI A. Optimizing Neural Networks for Efficient FPGA Implementation: A Survey[J]. Archives of Computational Methods in Engineering, 2021, 28: 4537-4547.

[24] CHEN Y R, WANG Y T. A survey of architectures of neural network accelerators[J]. Science China Information Sciences, 2022, 52:16.

[25] CHAKRADHAR S T, SANKARADASS M, JAKKULA V, et al. A dynamically configurable coprocessor for convolutional neural networks[C]// 37th International Symposium on Computer Architecture(ISCA 2010). Saint-Malo, France, 2010.

[26] ZHAO R, NIU X, WU Y, et al. Optimizing CNN-Based Object Detection Algorithms on Embedded FPGA Platforms[C]// 13th International Symposium on Applied Reconfigurable Computing (ARC). 2017:255-267.

[27] GOKHALE V, JIN J, DUNDAR A, et al. A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks[C]// 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014:696-701.

YU Yunjun, born in 1978, Ph.D, associate professor. His main research interests include fault diagnosis, active disturbance rejection control (ADRC), data-driven optimal control and its applications in microgrids, and low-carbon electricity technology.